MASTER EN SCIENCES INFORMATIQUES

MANET-Sim

Un simulateur de Modèle de Réputation et/ou de Confiance en environnements MANETs

Blangenois, Jonathan

Award date:
2013

Awarding institution:
Universite de Namur

Facultés Universitaires Notre-Dame de la Paix, Namur
Faculté d'Informatique
Année académique 2012–2013

MANET-Sim : Un simulateur de Modèle de Réputation et/ou de Confiance en environnements MANETs

Jonathan Blangenois

Maître de stage : Christophe Feltus
Promoteur : Michaël Petit

Mémoire présenté en vue de l'obtention du grade de
Master en Sciences Informatiques.

Résumé

Au cours de notre décennie, l'apparition en grand nombre de terminaux mobiles à relancer l'intérêt pour les architectures agents. Reprises sous l'acronyme MANET, cette dernière évolution des réseaux ajoutant la mobilité aux agents rencontre une popularité en hausse grâce à de nombreux cadres d'applications. Néanmoins les agents embarquant des ressources limitées, ils ne peuvent avoir recours qu'à des mécanismes allégés de sécurité. Pour combler ces manques et garantir un haut niveau de qualité de service, les modèles de réputation et/ou de confiance s'imposent comme une solution viable à ce problème. Associé à ceux-ci des simulateurs d'environnements d'utilisations pour modèle permettent de tester et de comparer les modèles dans les différents environnements excepté pour les MANETs pour lesquels il n'existe encore aucun simulateur. Pour remédier à ce problème, nous présentons notre propre plate-forme MANET-Sim capable de simuler les modèles dans les environnements MANETs, P2P, WSN et MAS.

Mots-clès : MAS, MANET, P2P, WSN, Modèle, Réputation, Confiance, Simulateur.

Abstract

During the last 10 years, interest in multiagent architecture with mobile devices network increase rapidly. Called MANET, this last evolution of network with mobility became rapidly popular, due to many use cases. However agents suffer from limited resources and can only use lightweight security mechanisms. To fill this gap and ensure a high level of quality of service, trust and/or reputation models are emerging as a viable solution to this problem. To compare and test models between them, simulation frameworks have been made for each type of networks, excepted for MANET for which there is actually no existing simulator. In order to solve the problem, we present MANET-Sim, our own simulation framework for trust and/or reputation models in MANET, P2P, WSN and MAS network.

Keywords : MAS, MANET, P2P, WSN, Model, Reputation, Trust, Framework, Simulation.

Avant-propos

Au travers de ce travail je tente de présenter une partie des recherches réalisées durant mes 5 mois de stage de fin d'études au *Centre de Recherche Public Henri Tudor* (Luxembourg) dans le domaine des systèmes multi agents et des modèles de réputation et/ou de confiance. Au cours de celui-ci je présente le cheminement réalisé pour la conception d'un simulateur de modèle de réputation et/ou de confiance dans les environnements *MANETs* et met en avant les bénéfices apportés par ces simulateurs pour la recherche (amélioration et études des modèles) ou pour l'aide apportée aux concepteurs de réseaux dans le choix d'un modèle (aide à la personnalisation et à la comparaison).

Je tiens par ce travail à remercier le *CRP Henri Tudor* et plus particulièrement les personnes qui m'ont encadré tout au long de mon stage : mon maître de stage *Christophe Feltus*, *Djamel Khadraoui* ainsi que *Guy Guemkam* qui ont crues en moi et m'ont permis de participer activement à leurs projets ainsi qu'à trois publications, dont une en temps que principal auteur.

Je tiens aussi à remercier mon promoteur *Michaël Petit* pour sa disponibilité, qu'elle soit physique ou virtuelle (email, Skype) tout au long du projet et pour son aide pour diriger le mémoire sur un seul (et unique) sujet du stage.

Table des matières

Chapitre 1

Introduction

Au cours du 21éme siècle l'intérêt porté par les constructeurs et les chercheurs concernant les architectures agents n'a cessé d'augmenter proportionnellement aux ouvertures des marchés en réponse aux nouveaux besoins orientés-service. Initié et reconnu d'abord par les systèmes multiagents centralisés et les réseaux pair-à-pair (P2P) décentralisés pour leurs efficacités, il aura fallu attendre encore une décennie et l'arrivée des senseurs ainsi que leur démocratisation pour constater un regain de l'intérêt porté aux architectures agents. Néanmoins là où les classiques mécanismes de sécurité comme le chiffrement pouvaient être correctement exploité dans les réseaux standard, les agents et les senseurs par leur plus faible caractéristique technique en terme de puissance de calcul, de mémoire et d'autonomie remettent à mal ces mécaniques ayant fait depuis longtemps leurs preuves et demandent la mise en place de nouveau mécanisme de sécurité plus allégé.

Pour répondre à ces besoins, la communauté scientifique s'est tournée vers les modèles de réputation et/ou de confiance. Initié par S. Marsh en 1994, le concept des modèles de réputation et/ou de confiance est de fournir à l'agent une aide décisionnelle en se basant sur l'analyse de précédentes interactions. Ainsi l'agent va au cours de sa vie enregistrer les résultats de ses interactions avec d'autres agents sous la forme d'informations pour ensuite les réutiliser pour déterminer s'il continue à collaborer avec ces mêmes agents. Dès lors un agent se comportant mal sera au fur et à mesure des interactions reconnu comme étant malicieux et ne sera progressivement plus sollicité.

Les modèles de réputation et/ou de confiance ont depuis S. Marsh fortement évolué par leur mécanique passant d'algorithme analytique à la logique floue ou notamment par l'exploitation des réseaux bayésien. À ces différents mécanismes, les chercheurs sont aussi venus ajouter de nouvelles fonctionnalités aux modèles comme la capacité d'obtenir de l'information directement auprès d'autres agents et de partager celle-ci. La communauté

a aussi commencé à doter ses modèles de la capacité de prévoir, sur base de statistiques, les comportements de certains agents et de les équiper de procédures et de méthodes de détection de comportements anormaux de plus en plus complexes. Ainsi au fil du temps les modèles se sont autant étoffés que diversifié et présentent chacun des capacités et des attributs différents des uns des autres.

Toutefois, cette pléthore de modèle et leurs diversités rend ardue la tâche des concepteurs de réseaux d'agents dans le choix d'un modèle. Pour y remédier la communauté scientifique à réalisé la conception de simulateurs de modèle de réputation et/ou de confiance dans les 3 grands types de réseaux d'agents reconnus que sont les réseaux multiagents, les réseaux P2P et les réseaux de senseurs sans fil. Ainsi aidés de ces outils les concepteurs de réseaux d'agents ont pu comparer des modèles entre eux grâce à des métriques collectées durant les simulations, tester différentes combinaisons de paramètres pour un même modèle pour les adapter à leurs cas d'utilisations ou encore de confronter leur topologie à différentes catégories d'agents malicieux.

Néanmoins une dernière évolution de ces réseaux, poussée par l'adoption massive d'appareil intelligent tels que les smartphones et par de nouveaux besoins découlant d'utilisation de nouvelles technologies, est venue ajouter l'aspect de mobilité des agents aux trois grands réseaux existants. Cet ajout de la mobilité impliquant une topologie du réseau en perpétuelle évolution et demandant aux agents de constamment maintenir à jour leurs informations de routage. Regroupé sous l'acronyme *MANETs*[1] ces nouveaux réseaux mobiles sont depuis peu sous le feu des projecteurs par exemple avec des projets comme le *Google Loon*, initié par la firme *Google* dont le but est d'offrir une couverture internet à des zones non desservies grâce à un maillage mobile de ballons envoyés dans la stratosphère. D'autres projets *MANETs* comme de l'échange d'informations de trafic entre véhicules sont notamment aussi portés par de nombreux consortiums et universités.

Les *MANETs* se positionnent finalement comme la nouvelle évolution phare des réseaux des prochaines années à venir et avec eux les architectures agents embarquant des modèles de réputation et/ou de confiance. Néanmoins il n'existe pas actuellement de simulateur de modèle de réputation et/ou de confiance implémentant les aspects de mobilité. Il parait alors légitime de s'interroger sur les capacités des simulateurs existants et leur possible intégration des aspects de mobilité.

Au cours de ce travail nous explorerons au chapitre 1 le monde des modèles de réputation et/ou de confiance en découvrant leurs fonctionnements et leur différent environnement utilisations. Cette exploration nous permettra de comprendre les différents besoins et spécificités des modèles et de leurs environnements d'utilisations et de confondre

1. Mobile Ad-hoc NETwork

ceux-ci aux capacités de simulation de différent simulateur existant. Nous pourrons ainsi découvrir les manquements de ceux-ci pour la simulation des *MANETs* et envisager une solution au chapitre 2 sous la forme d'une modification d'un simulateur existant ou la conception d'un nouveau simulateur compatible *MANETs*.

Chapitre 2

État de l'art

Nous débuterons l'état de l'art par présenter les principes généraux de fonctionnement des modèles de réputation et/ou de confiance pour ensuite discuter les notions de confiance et de réputation et des différents environnements d'utilisations de ceux-ci. En seconde partie nous illustrerons les différents comportements malicieux les plus représentatifs pour ensuite présenter une collection de modèles de confiance et/ou de réputation et illustrerons leur variétée au travers de leur différents mécanisme et environnement d'utilisation. Nous enchainerons ensuite par une exploration du monde de la simulation des modèles de réputation et/ou de confiance par une classification des métriques de comparaison de modèles et une analyse de simulateur existant.

2.1 Principes généraux des modèles de réputation et/ou de confiance

Le fonctionnement général d'un modèle repose sur l'interrogation d'un individu A par rapport à un individu B fournissant un service S dont A pourra retirer un niveau de satisfaction ϕ. L'enjeu pour A est de déterminer si le service fournit par B lui procurera un niveau de satisfaction à la hauteur de ses attentes.

Pour ce faire l'individu A va réaliser une succession de phase présentée à la figure 2.1 et que nous détaillons :

1. **Collecte d'information** : La première phase consiste à collecter de l'information sur B. Ces informations peuvent être symbolisées par des valeurs déjà calculées (ex : réputation, confiance ...) ou par des collections de précédentes évaluations. Les évaluations représentent des niveaux de satisfaction générée par les individus à partir de services rendus par B. Toutes ces informations peuvent généralement

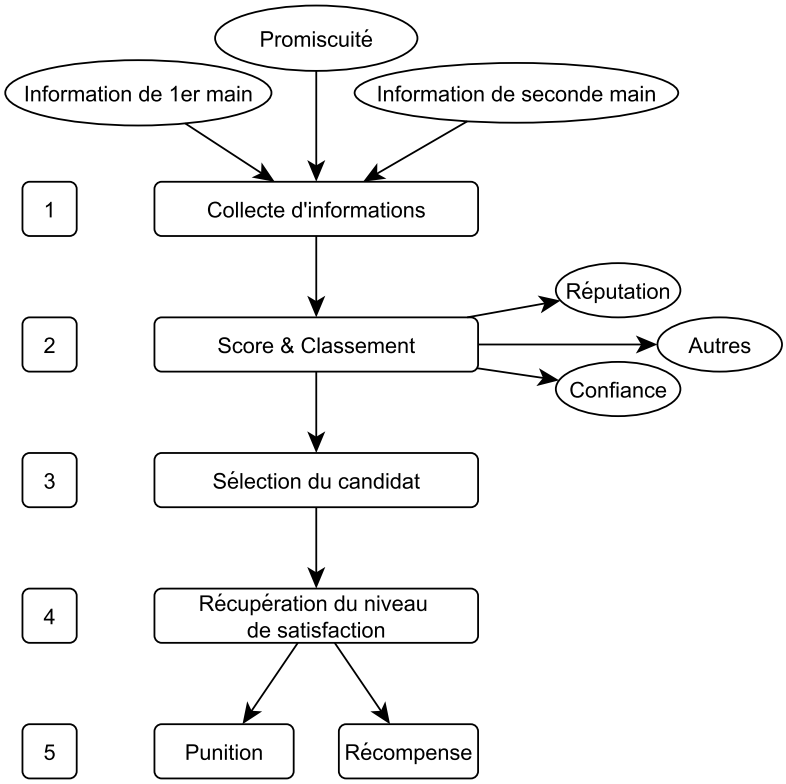

FIGURE 2.1: Phase de fonctionnement générale d'un modèle

être obtenues suivant trois sources :

- De manière directe : A est l'auteur des informations relatives à B. Nous parlerons des informations acquises de manière directe comme étant des **informations de première main**.
- De manière indirecte : A collecte des informations portant sur B auprès d'autres individus. Nous identifierons les informations acquises de manière indirecte comme étant des **informations de seconde main**.
- Par procuration : A est témoin d'informations en lien avec B (ex : obtenu par promiscuité dans un réseau sans fil). Nous identifierons les informations acquises par procuration comme faisant partie des informations de première main.

2. **Score et Classement** : En seconde phase A va agréger, selon les mécanismes du modèle qu'il exécute, les informations collectées durant la première phase pour obtenir un ou plusieurs scores (niveau de confiance, valeur de réputation, fiabilité ...) qu'il attribuera à B. Ces scores seront analysés par le modèle qui prendra la

décision de classer B sur base de critères propres à lui même, citons par exemple :

- Digne ou non digne de confiance : suivant un score calculé pour B, celui-ci sera comparé à un seuil permettant de déterminer si B est digne ou indigne de confiance.
- Malicieux ou non malicieux : L'analyse des scores et des informations collectées sur B permettront à certain modèle de détecter des comportements anormaux.

Notons que la nature d'un score peut se présenter sous différente forme, par exemple de manière binaire (ex : digne de confiance, indigne de confiance) ou sur base d'intervalles (ex : [0,1], [1,10]). [26] met en alerte sur les différences de granularité dans le calcul des scores en mettant en avant le fait que de nombreux modèles attribuent des scores uniques pour un ensemble de service fournit alors que d'autres modèles attribuent un score propre pour chaque service fournit par un individu.

3. **Selection du candidat** : La troisième phase consiste à la sélection d'un candidat. Si dans notre exemple B est le seul candidat, il est de nature générale que plusieurs candidats puissent correspondre à la recherche de service de A. Dans ce cas de figure, la majorité des modèles choisiront le candidat possédant le plus haut score tout en écartant les candidats qui seront jugés indignes de confiance ou malicieux. Notons qu'il est tout à fait envisageable qu'une sélection de candidats soit entièrement rejetée par le modèle (ex : indigne de confiance).
4. **Récupération du niveau de satisfaction** : La quatrième phase débute après la sélection d'un candidat et la réalisation du service par celui-ci. Cette réalisation du service par le candidat B va générer auprès de A un niveau de satisfaction. Celui-ci pouvant être de nature binaire (positif, négatif) ou débitaire d'une fonction de satisfaction. Le niveau de satisfaction ainsi obtenu sera comparé à un seuil permettant de déterminer l'action entreprise par A en phase 5.
5. **Récompense et Punission** : La dernière et cinquième phase est déterminé par le niveau de satisfaction obtenu en phase 4. Comparé à un seuil, celui-ci va déterminer une action de récompense ou de punition de A envers B. Ces actions se traduisant par l'enregistrement d'information par A symbolisé par une nouvelle évaluation (ex : binaire, intervalle ...) portant sur B ou par la modification de valeurs déjà calculée (ex : pondération, incrémentation/décrémentation ...).

2.2 Concepts de Confiance et de Réputation

[30] nous apprends que les concepts de *Confiance* (Trust) et *Réputation* (Reputation) sont depuis longtemps étudiés dans des domaines comme la psychologie, la sociologie ou

encore l'économie. Cependant comme celui-ci, nous concentrerons notre analyse sur le domaine des sciences informatiques et plus particulièrement l'intelligence artificielle qui a connu un essor grâce à l'émergence des systèmes multi agents et des sites de e-commerce. Dans cette section nous montrerons que les concepts de *Confiance* et de *Réputation* émane d'interprétation propre à chacun des auteurs de modèles et que ceux-ci reflètent l'opinion d'un individu par rapport à un autre. Leurs différences provenant des sources d'informations utilisées pour calculer la valeur d'opinion. Nous terminerons cette section par la présentation des différents environnements d'utilisation dans le domaine de l'IT [1].

2.2.1 Définition des concepts de Confiance et de Réputation

Dans son analyse [14] définis la *Confiance* dans le cadre d'un système IT comme *"étant une probabilité subjective avec laquelle un individu A prévoit qu'un individu B réalise une action dont son bien-être dépend"* [2]. Cette définition fait figurer la Confiance en termes de probabilité et induit que le fait de faire confiance à quelqu'un place l'individu dans un état de dépendance (et donc de vulnérabilité) envers l'autre, offrant la possibilité d'être affaibli.

Néanmoins [10] affirme que la confiance est plus qu'une *"probabilité subjective"* [3] et avance l'argument que posséder une forte probabilité qu'un individu réalise une action n'est pas suffisant pour dépendre de celui-ci. Partant de ce constat, ils tentent de présenter la *Confiance* sous l'aspect d'un *"ingrédient mental"* lié à la croyance (Belief), aux buts à réaliser (Goals to achieve) et à la décision de déléguer (acceptation de dépendre d'un autre individu pour la réalisation d'un service). Cette approche sous-entends que seuls les agents possédant des buts et des croyances peuvent avoir confiance en d'autres agents. Ainsi si nous prenons deux agents A et B, le fait que A ait confiance en B selon [10] implique que temporairement les buts de A deviennent ceux de B.

Une autre approche est proposée par [20] statuant que la Confiance *"est l'importance avec laquelle un parti donné est volontaire pour dépendre de quelque chose ou de quelqu'un dans une situation de sentiment relatif de sécurité, malgré les risques de conséquences négatives."* [4].

La définition proposée par [18] définie la réputation comme étant "*une mesure dérivé à partir de la connaissance d'interactions directes et/ou indirectes passées, et utilisée*

1. Information technology
2. *"Trust is the subjective probability by which an individual, A, expects that another individual, B, performs a given action on which its welfare depends."*
3. *... is much more than subjective probability.*
4. *"Trust is the extent to which a given party is willing to depend on something or somebody in a given situation with a feeling of relative security, even though negative consequences are possible."*

pour accéder au niveau de confiance qu'un agent porte à un autre." [5] . La confiance quant à elle est définie par ce même auteur comme étant la probabilité effective qu'un agent se comporte comme espérée.

Quant aux auteurs de [26], ceux-ci définissent la réputation comme *une attente à propos du comportement d'un agent basé sur des informations ou des observations de ses précédents comportements.* [6] et retiennent la confiance comme étant *un niveau particulier d'une probabilité subjective en lien avec la réalisation d'une action particulière par un agent, à la fois avant que celui-ci puisse contrôler cette action et dans un contexte influençant sa propre action* [7]

Notons que de nombreux auteurs ne font pas preuve de distinction et de précision lors de leur utilisation des concepts de *Confiance* et/ou de *Réputation* alors que d'autres séparent clairement les deux concepts et qu'il existe presque autant de définitions pour ces deux concepts qu'il existe de modèle. Tout au long de ce travail, nous ferons le choix de considérer la *Confiance* et la *Réputation* comme étant des valeurs reflétant l'opinion d'un individu, calculé suivant une ou plusieurs mécaniques, par rapport à la capacité d'un autre individu à fournir un service. Toutefois, nous continuerons à utiliser les termes de *Confiance* et de *Réputation* pour distinguer les types de sources d'informations utilisés lors du calcul de ces valeurs d'opinion. Ainsi nous représenterons :

- La *Confiance* comme étant une valeur issue des informations de première main.
- La *Réputation* comme étant une valeur issue des informations de première et de seconde main.

Entre autres nous distinguerons lors du calcul d'une valeur de *Réputation* l'origine des évaluations. Nous utiliserons le terme de *Réputation direct* (DR_{ab}) [8] pour les valeurs de réputation calculée à partir d'information de première main que possède un agent A par rapport à un agent B, ainsi que le terme de *Réputation indirecte* (IR_{cb}) [9] pour les réputations calculées à partir d'informations de seconde main fournit par un voisin C concernant un agent B. Nous appellerons *Réputation globale* (OR_{ab}) [10] une réputation calculée à partir de l'association des informations de première et de seconde main par

5. *"Reputation is a measure that is derived from direct and/or indirect knowledge of earlier interactions if any, and is used to access the level of trust an agent puts into another."*
6. *"A reputation is an expectation about an agent's behavior based on information about it or observations of its past behavior"*
7. *"Trust (or, symmetrically, distrust) is a particular level of the subjective probability with which an agent will perform a particular action, both before he can monitor such action (or independently of his capacity ever to be able to monitor it) and in a context in which it affects his own action."*
8. Direct Reputation
9. Indirect Reputation
10. Overall Reputation

l'agent A sur l'agent B. Nous appellerons *recommendeur* un tiers nous procurant des informations de seconde main. La confiance, quand à elle, sera symbolisé par le symbole T.

2.2.2 Environnements d'utilisation

Dans ce travail nous présenterons les environnements d'utilisation comme étant constitués des systèmes multi agents (MAS [11]), des réseaux Peer-to-peer (P2P), des réseaux de senseur sans fil (WSN [12]) et des réseaux mobiles ad-hoc (MANET [13]). Nous présentons dans cette section chacun d'entre eux et fournissons des exemples d'utilisation des concepts de *Réputation* et de *Confiance* présentés au point précédant.

Systèmes Multi Agent (MAS)

Les systèmes multi agent sont des systèmes constitués d'agents et de leur environnement. Les agents peuvent être des agents logiciels comme ils peuvent être des robots, des humains, des groupes d'humains ou encore une combinaison de plusieurs types. Ces systèmes sont organisés en fonction des objectifs qu'ils tentent d'atteindre et répondent à plusieurs propriétés :

- L'autonomie : l'agent est au minimum partiellement autonome.
- Vue locale : aucun agent du système ne possède de vue globale du réseau.
- Décentralisé : le système n'est pas un bloc monolithique.

Les systèmes multi agent sont référencés comme étant des systèmes "auto-organisés" dont le but est de solutionné leur problème sans "intervention extérieure". Leur principale particularité est leur modularité permettant d'étendre ou de reconstruire un système sans devoir réécrire une partie de l'application des agents logiciel. Entre autres ces systèmes peuvent rapidement se remettre d'une panne grâce à la forte redondance des composants et à l'autogestion des fonctionnalités. L'utilisation des concepts de *Réputation* et de *Confiance* permet d'augmenter la réactivité au sein du système dans le cadre de la détection de panne et d'apporter un aspect proactif permettant d'écarter du système les agents montrant des signes de faiblesses ou de comportements anormaux. Les systèmes multi agent profitent d'une architecture centralisée basée sur des hiérarchies entre les agents pour la transmission d'informations. Ainsi un agent de niveau 2 recevra de l'information de plusieurs agents de niveau 1 qu'il d'agrégera avant de les envoyer à un agent

11. Multi-Agent System
12. Wireless Sensor Network
13. Mobile Ad-hoc NETwork

de niveau supérieur. Cette opération se répétera sur l'ensemble des niveaux jusqu'à atteindre un agent centralisant l'ensemble de l'information du réseau. Notons les réseaux de type multi agent pour être de nature filaire, sans-fils ou encore hybride et être sous la surveillance d'un superviseur chargé de gérer le réseau.

Dans son article [6] présente l'utilisation d'un réseau multi agent dans le cadre de gestion d'infrastructure bancaire auquel l'auteur fait embarquer à chaque agent un modèle de réputation et/ou de confiance afin d'augmenter la réactivité du réseau à la détection de panne ou de fraude. [1] et [2] se situent dans le cadre du e-commerce et présentent des systèmes multi agents humains/logiciels dont le rôle grâce aux concepts de *Confiance* et de *Réputation* est de fournir au système un mécanisme de détection dissuasif pour les tricheurs et les fraudeurs au sein de leur site internet respectif Amazon et eBay.

Réseaux Peer-to-peer (P2P)

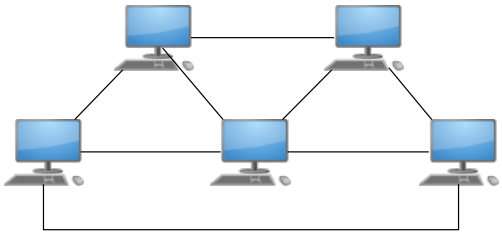

FIGURE 2.2: Illustration d'un réseau Peer-to-peer.

Les réseaux de type Peer-to-peer (Figure 2.2) sont des réseaux dans lesquels chaque participant peut jouer le rôle de client et de serveur dans le but de partager des accès à des ressources diverses comme des fichiers, des périphériques ou des capacités de calcul sans le recours à un serveur central.

[3] nous apprend qu'un système P2P possède plusieurs propriétés :

1. Aucune coordination centrale.
2. Pas de base de donnée centrale.
3. Aucun participant ne possède de vue globale du système.
4. Chaque participant est autonome.
5. Les participants et les connexions ne sont pas fiables (ad-hoc).

Dans son analyse des réseaux P2P créée par l'application Gnutella, [4] mets en évidence le problème d'équilibrage des charges dans le réseau dû à la volonté inexistante

de coopération de certains participants (conséquence de la propriété 5). [3] Repart de ces constatations pour justifier l'utilisation des concepts de *Confiance* et de *Réputation* au sein d'un réseau P2P arguant la nécessité de ces concepts pour chaque cas possible d'utilisations allant du téléchargement d'un fichier, où l'on souhaiterait sélectionner uniquement des participants fiables, au cas plus complexe d'une communauté jouant le rôle d'une place de marché où la *Confiance* entre chaque participant est un prérequis pour la réalisation de transaction. Les réseaux P2P sont perçus comme étant des systèmes multi agent décentralisé.

Réseaux de Senseur sans fil (WSN)

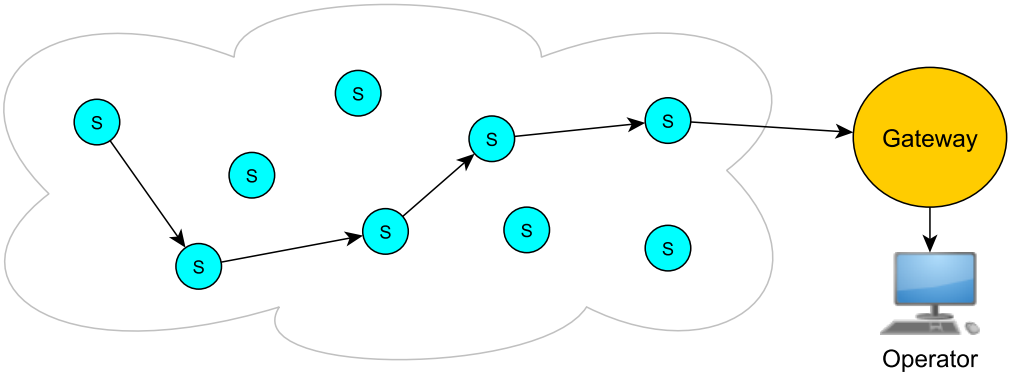

FIGURE 2.3: Illustration d'un réseau de senseur sans fil.

Les réseaux de senseurs sans fil (Figure 2.3) sont des réseaux partiellement distribués dans lesquels une collection de senseurs autonome réalise la supervision d'appareils physique ou de conditions climatiques comme la température, la vitesse du vent ou encore l'humidité. L'ensemble des senseurs du réseau travaille de concert pour faire remonter les données collectées auprès d'un senseur principal (Gateway) dont le rôle est d'agréger les données reçues et de déterminer certaines caractéristiques de l'environnement ou de détecter des événements. [7] nous apprend que les senseurs sont de petits appareils produits à faible coût embarquant des batteries à capacité limitée leur permettant d'être largement déployé dans différents domaines allant de la surveillance militaire à la gestion de trafic routier. Néanmoins l'une des problématiques propres à ce type de réseau ,en plus des risques liés aux réseaux sans fil, vient de la composante des senseurs et tout particulièrement de leurs faibles caractéristiques en termes de bande passante, de puissance

de calcul, de capacité de mémoire et de batterie empêchant l'application des traditionnels mécanismes reconnus de confidentialité et d'authentification due de leur trop grande consommation en ressources. C'est ainsi que l'utilisation des concepts de *Confiance* et *Réputation* a été proposé pour pallier ce manquement et offrir aux senseurs de l'aide dans la réalisation de choix (decision-making) et promouvoir la collaboration entre les senseurs fiables tout en prenant en compte les contraintes de coût énergétique et de rapidité d'exécution.

Réseaux mobiles Ad-hoc (MANET)

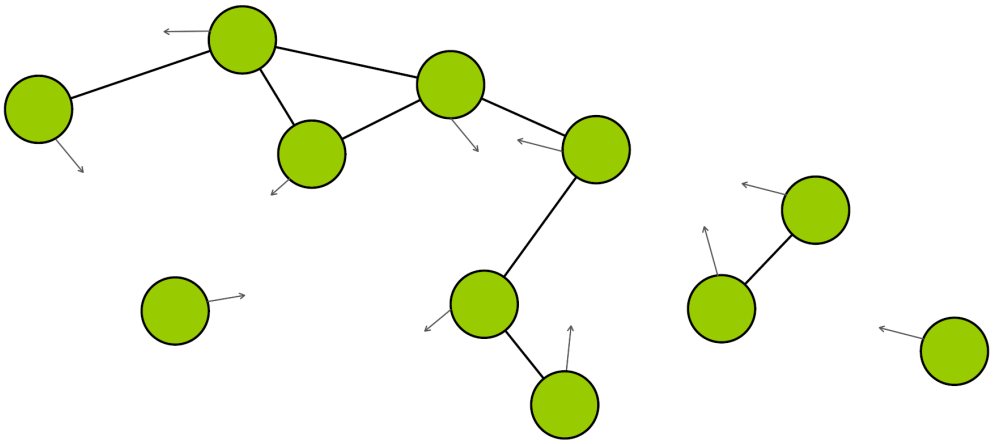

FIGURE 2.4: Illustration d'un réseau mobile ah-hoc sans fil.

Les réseaux mobiles ad-hoc (Figure 2.4) sont considérés comme des infrastructures auto configurée d'appareils mobiles connectés entre eux via par une liaison sans fil. L'une de leurs grandes particularités est due à l'aspect mobile des appareils induisant une topologie du réseau en constante modification et imposant aux appareils de constamment maintenir à jour leurs informations de routage. [11] nous montre que les appareils évoluant dans un *MANET* souffrent de contraintes moins fortes que les senseurs des *WSN* en termes de ressources (puissance de calcule, énergie, bande passante, temps de réaction), mais souffrent en plus de contraintes dynamiques (changement de topologie, mobilité, disparition d'agent ...). [18] précise que la qualité de service d'un MANET dépend essentiellement du niveau de collaboration entre les différents appareils, ceux-ci faisant office de relais lors de l'échange de paquet ils peuvent à tout instant choisir d'arrêter de collaborer. Les concepts de *Confiance* et de *Réputation* sont utilisés dans les *MANETs* pour solutionné les contraintes de ressource (similaire aux *WSN*), mais

aussi pour appuyer les décisions de collaboration permettant entre autres de détecter les appareils qui pourraient refuser de collaborer, le tout dans une optique de préservation d'énergies et de ressources. Les réseaux de type MANETs couvrent plusieurs autres catégories de réseaux mobiles notamment :

- VANET : Les VANETs [14] dont les agents sont représentés par des véhicules s'échangeant de l'information sur le trafic routier.
- SMANET : Les SMANET [15] Représente des réseaux de senseurs mobiles.
- iMANET : Les iMANET [16] avec notamment le projet *Google Loon* [17] lancé en 2013 dont le but est de réaliser la couverture internet de zone non desservie grâce à la mise en place d'un maillage à partir de ballon aérien mobile.

2.3 Comportements malicieux

Dans cette section nous présenterons les comportements malicieux les plus communs réalisés par des agents dans les différents environnements présentés en 2.2.2. Par comportement malicieux nous entendons des comportements intelligents et motivés dont l'impact est de perturber le réseau et qui sont potentiellement détectables par les modèles de réputation et/ou de confiance grâce à des mécaniques d'analyse des comportements (ex : sur base des évaluations). Ces comportements sont réalisés par des agents du réseau soit dans un but de nuisance soit à cause de contrainte technique (ex : un niveau de batterie faible). La sélection de comportements malicieux que nous présentons dans cette section se base sur [23] ainsi que sur les différents articles liés aux modèles présentés en section 2.4. Au cours de celle-ci nous spécifierons les origines possibles des comportements malicieux ainsi que leurs impacts sur le système et nous les illustrerons dans un environnement sans fil avec promiscuité dans un simple contexte de services de relays de paquets. L'ensemble des comportements présenté est naturellement transposable dans l'ensemble des environnements présentés en 2.2.2 ainsi qu'à tous autres types de service, la promiscuité n'ajoutant que des effets collatéraux aux comportements originaux.

Entre autres nous illustrerons la détection des malicieux par l'utilisation d'un modèle simple de confiance que nous inventons à cet effet. Notre modèle d'illustration que nous nommerons $DEMO$ possedera un modèle de décision simple consistant à simplement soustraire le nombre d'évaluation négative (β) au nombre d'évaluation positive (α) pour obtenir un score de confiance T. Si celui-ci est positif ($T \geq 0$) l'agent évalué sera considéré

14. Vehicule Ad-hoc NETwork
15. Sensor Mobile Ad-hoc NETwork
16. Internet Mobile Ad-hoc NETwork
17. http ://www.google.com/loon/

comme étant digne de confiance. A l'inverse, si le score de confiance est négatif, l'agent évalué sera considéré par notre modèle comme étant indigne de confiance et malicieux.

2.3.1 Selfish

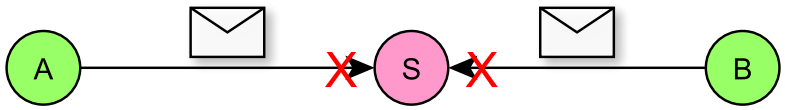

FIGURE 2.5: Comportement malicieux de type Selfish

Le comportement malicieux de type *Selfish* (égoïste) est un comportement basique durant lequel un agent refuse catégoriquement de realiser les services demandé par d'autre agent du réseau tout en continuant à utiliser le système pour réaliser les services dont il a besoin. L'illustration en figure 2.5 montre un simple réseau composé de trois agents dont l'agent S en position centrale est un agent égoïste qui empêchera les paquets allant de A vers B et inversement. La décision d'un agent de devenir égoïste par rapport au système peut être soit d'origine (dans un but de perturbation), soit motivée par des contraintes propres à l'agent, par exemple, un niveau de batterie faible qui lui imposerait une réduction de sa collaboration au sein du système pour se préserver. Le blocage créé par un comportement égoïste impose aux environnements décentralisés de découvrir de nouvelle route pour contourner l'obstacle et peut suivant la topologie du réseau isoler des agents ou des groupes d'agents de l'accès à un ou plusieurs services. Néanmoins leur détection par les modèles est généralement efficace, car leur nombre d'évaluations négatives augmentera au fur et à mesure qu'ils refuseront de collaborer, réduisant ainsi au fil du temps les valeurs de confiance et/ou de réputation . Précisons aussi qu'un agent qui détecte son voisinage comme étant malicieux ne réalisera plus d'interaction avec celui-ci.

L'illustration avec notre modèle d'exemple $DEMO$ executé par l'agent A pour évaluer la réalisation de 100 services par S donnera un score de confiance négatif ($\alpha = 0, \beta = 100$) et donc une détection de l'agent S comme étant un malicieux.

2.3.2 On/Off

Un agent réalisant un comportement malicieux de type *On/Off* réalisera un même service réellement une fois sur deux auprès des autres agents du réseau. Illustré dans la fi-

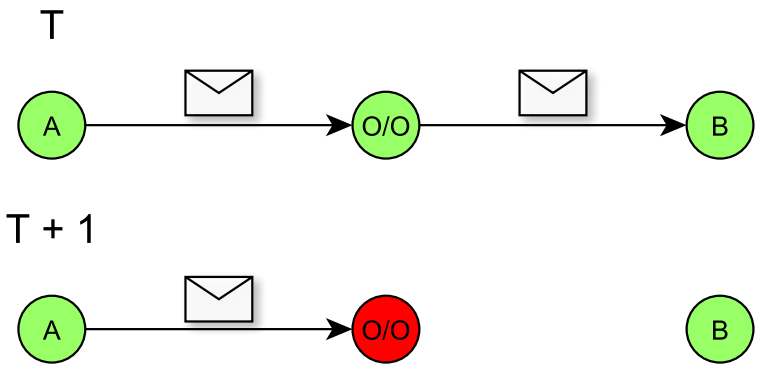

FIGURE 2.6: Comportement malicieux de type On/Off

gure 2.6, l'agent *On/Off* collabera avec l'agent A à la demande de service T et transférera le paquet à l'agent B, mais refusera de collaborer avec A à la demande de service $T+1$, bloquant ainsi 50% des paquets de l'agent A à destination de B. La problématique apportée par les comportements de type *On/Off* est qu'au cours de leur interaction avec le système le nombre d'évaluations positive et négative d'un agent *On/Off* sera similaire, amenant dans le cas de modèle attribuant le même poid à une évaluation positive et négative, la création d'un équilibre entre les deux valeurs et de ce fait une non-détection de l'agent comme étant malicieux.

L'illustration avec notre modèle d'exemple $DEMO$ executé par l'agent A pour évaluer la réalisation de 100 services par O/O donnera un score de confiance nulle ($\alpha = 50$, $\beta = 50$) et donc une non-détection de l'agent O/O comme étant un malicieux par notre modèle.

2.3.3 Oscillant

Les comportements de type *Oscillant* regroupent l'ensemble des comportements malicieux similaires aux comportements de type *On/Off* mais avec des périodes plus longues de réalisation de service avant un refus de réalisation de service. Les comportements *On/Off* étant des cas particuliers d'*Oscillant* sur une période de 1 et les *Selfish* des *Oscillants* avec une période de 0. La difficulté de détection des comportements *Oscillants* augmente proportionnellement par rapport à la période n (Figure 2.7) de collaboration de l'agent avant refus, amenant en fonction de celle-ci le blocage de $\frac{100}{n+1}\%$ des interactions. De plus, plus celle-ci est grande et plus l'écart entre le nombre d'évaluations

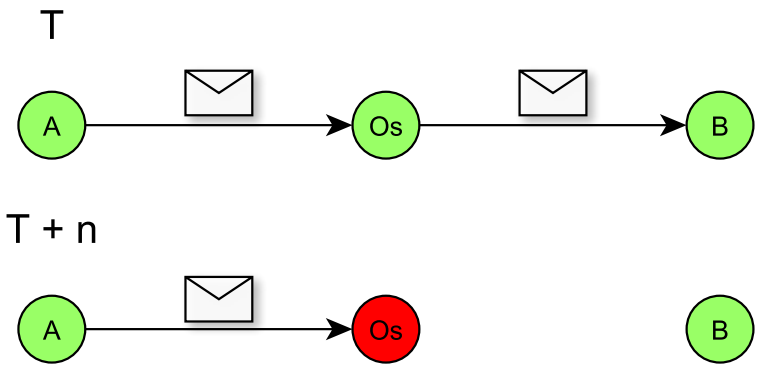

FIGURE 2.7: Comportement malicieux de type Oscillant

positive et négative est grand, rendant de ce fait plus difficile la détection de l'agent comme étant malicieux, car proche d'un comportement normal.

L'illustration avec notre modèle d'exemple $DEMO$ executé par l'agent A pour évaluer la réalisation de 100 services par Os donnera un score de confiance T ($\alpha = 100 \times (100\% - \frac{100}{n+1}\%)$, $\beta = 100 \times \frac{100}{n+1}\%$) qui sera positif pour toute période $n \geq 1$. Ainsi notre modèle ne sera capable que de détecter les agents malicieux oscillant ayant une période $n = 0$, c'est à dire uniquement les malicieux de type *Selfish*.

2.3.4 Bad-Mouthing

Les agents réalisant des comportements de type *Bad-Mouhting* (BM) fonctionnent par pair. Lorsque deux agents de type *Bad-Mouthing* se retrouvent, ceux-ci commencent à travailler ensemble pour réaliser leur comportement malicieux. Illustré en figure 2.8 dans un petit réseau sans fil avec promiscuité, le comportement malicieux s'exécute en deux phases :

- La première phase débute lors que l'agent A transfère un paquet à destination de D vers le premier agent de type *Bad-Mouthing* $BM1$. Celui-ci va alors transférer le paquet de l'agent A vers son camarade malicieux $BM2$. L'agent A constatant (dans notre exemple sans fil par promiscuité) que $BM1$ à correctement relayé son paquet va positivement l'évaluer (flèche verte).
- La deuxième phase va voir l'agent $BM2$ transférer la paquet vers un autre agent (dans notre exemple vers le destinataire D), cependant, et quelques soit l'action que réalisera l'agent choisi par $BM2$, celui-ci l'évaluera négativement (flèche rouge).

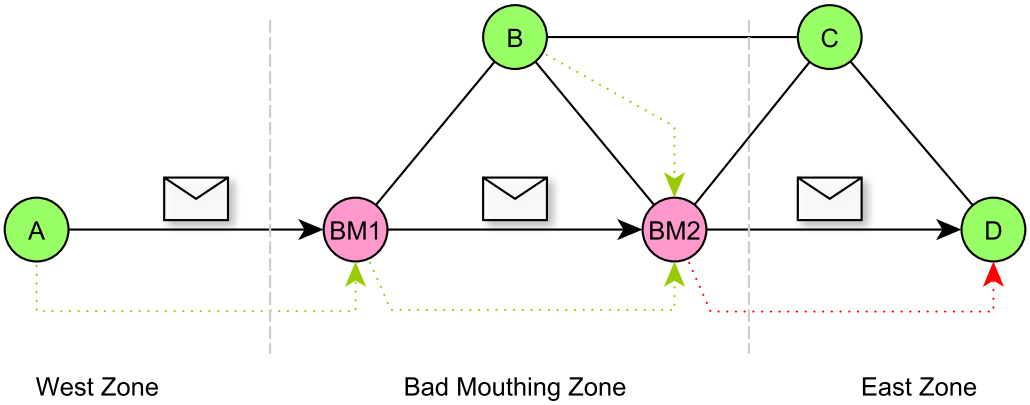

FIGURE 2.8: Comportement malicieux de type Bad-Mouthing en environnement sans fil

Entre autres comme $BM2$ à parfaitement relayé la transaction il sera évalué positivement par $BM1$ (et par les autres agents se trouvant en situation de promiscuité dans notre exemple sans fil, c'est à dire par l'agent B).

La présence d'une paire d'agents malicieux de type BM dans un système amène la création de frontières autour du couple de BM. Dans l'exemple présenté le comportement malicieux de la paire va amener à la frontière est une diminution de réputation de l'agent D par la création de l'agent $BM2$ de fausse évaluation négative. Si nous prenons l'exemple du calcul de la réputation de l'agent D par l'agent C, celui-ci ira collecter des informations de seconde main auprès du recommendeur $BM2$ qui feront baisser la réputation qu'il calculera pour D. Une fois que C décidera de ne plus faire confiance à D à cause des informations fournies par $BM2$, C s'intéressera à B qui sera aussi pollué par les fausses évaluations de $BM2$ amenant C à ne plus faire confiance qu'à $BM2$. L'impact sera le même lorsque les voisins de D ou B calculerons la réputation de C à cause de la pollution de $MB2$, amenant chacun à douter de ses voisins et à ne faire confiance qu'à lui et faisant de $BM2$ le prochain saut pour toutes leurs transactions permettant ensuite à $BM1$ de polluer la frontière ouest du système. Entre autres le travail en paire des deux agents BM permet de fédérer à leur cause d'autres agents qui se trouvent entre eux par exemple l'agent B qui ne pourra que constater que $BM1$ et $BM2$ transfère bien les transactions et leur apportera du poids auprès des autres agents grâces à sa collection d'évaluation positive sur chacun des deux.

Le but malicieux d'une paire de BM est de devenir au sein du système une zone obligatoire de passage pour toutes les transactions voisines et de par ce fait devenir un point stratégique du réseau. La grosse difficulté dans la détection des agents de types BM est que leur attaque se situe au niveau de la pollution des informations de seconde main provenant d'eux même sous la forme de fausse évaluation négative sur le voisinage, mais aussi de complices d'infortune qui fourniront aux autres agents des collections d'évaluations positives appuyant la fiabilité des agents de type *Bad-Mouthing*.

2.3.5 Ballot-Stuffing

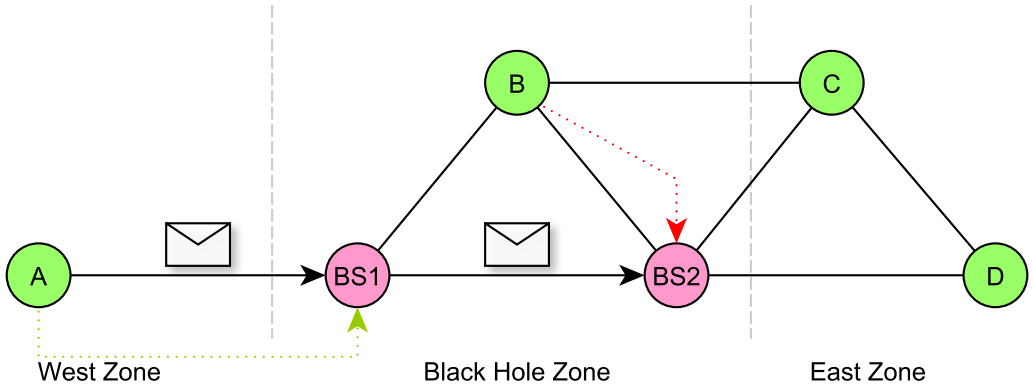

FIGURE 2.9: Comportemet malicieux de type Ballot-Stuffing en environnement sans fil

Similaire aux *Bad-Mouthing* les comportements malicieux de type *Ballot-Stuffing* (BS) se réalisent par paire et en deux phases (Figure 2.9) :

- La première phase débute lors que l'agent A transfère une transaction vers le premier agent de type *Ballot-Stuffing* $BS1$. Celui-ci va alors transférer la transaction de l'agent A vers son camarade malicieux $BS2$. L'agent A constatant (dans notre exemple sans fil par promiscuité) que $BS1$ à correctement relayé sa transaction va positivement l'évaluer (flèche verte).
- La deuxième phase va voir l'agent $BS2$ arrêter le transfert de la transaction (et être négativement évalué par les autres agents se trouvant en situation de promiscuité dans notre exemple sans fil, c'est à dire par l'agent B).

Dans le cas d'une paire d'agents de type BS le système sera divisé en frontière. Reprenant l'exemple de la figure 2.9 l'agent A va constater que l'agent malicieux $BS1$ a

correctement relayé sa transaction et attribuera à celui-ci une évaluation positive. Au fil des échanges l'agent $BS1$ va prendre de l'importance pour A qui va continuer à le considérer comme une valeur sûre pour ses transactions et $BS1$ transférera toujours à son complice qui stoppera les transactions de A. Les agents comme B se trouvant entre les deux agents malicieux pourront constater que $BS2$ ne transfèrent pas les transactions et lui attribuer des mauvaises évaluations qui permettront à des agents comme C d'obtenir des informations de secondes mains sur $BS2$. Le problème viendra de l'agent D qui n'ayant aucune information sur $BS2$ va lui transférer des transactions qui seront correctement relayées à $BS1$ qui lui les stoppera. L'agent D voyant que $BS2$ est un pair de confiance va continuer à réaliser des interactions avec lui et lui attribuer des évaluations positives qui entreront en conflit avec les évaluations négatives que fournit B lorsque l'agent C tentera de calculer la réputation de $BS2$ en questionnant son voisinage. Au final C pourrait se ranger sur l'opinion de D et considérer $BS2$ comme un saut potentiel pour ses transactions, une fois que C commencera à interagir avec lui, la réputation de $BS2$ augmentera encore et B sera le seul à considérer $BS2$ comme étant non fiable.

L'impact de la paire de BS dans le système sera d'être attractif pour les agents se trouvant aux frontières ouest et est, attirant ainsi les demandes d'interactions vers l'un ou l'autre des deux agents malicieux dans le but d'interrompre les transferts et de créer un trou noir au sein du système. Seuls les agents se trouvant à l'intérieur de celui-ci pourront constater ce qu'il en est, mais leurs recommandations ne feront pas longtemps le poids face à l'apparente respectabilité que les agents de type *Ballot-Stuffing* s'entraident artificiellement à augmenter et que les agents situés aux frontières constateront.

2.3.6 Flattering

Les agents de types *Flattering* (flatteur) sont des agents non objectifs qui évalueront toujours un voisin positivement qu'il relaye ou non la transaction. L'impact de ces agents est de polluer le système d'évaluation positive induisant en erreur les agents sur la qualité du réseau et de leur voisinage.

2.4 Modèles de réputation et/ou de confiance

Au cours de la dernière décennie le nombre de modèles de réputation et/ou de confiance a littéralement explosé mettant en évidence un intérêt pour l'application de ceux-ci dans les différents environnements d'utilisation présentée en 2.2.2. Le but de cette section n'est pas réaliser un état de l'art de tous les modèles de réputation et/ou de confiance existant, mais de fournir au lecteur un aperçu des principaux courants de

modélisation de ce domaine. Pour ce faire, nous réalisons une présentation d'une sélection de modèles classés selon les méthodes et techniques utilisées (logique floue, Bayésien, Analytique, Comportement biologique) pour le calcul des valeurs de réputation et/ou de confiance. Pour chacun d'entres eux, des informations seront spécifiées sur les stratégies du modèle pour, par exemple, gérer la crédibilité des informations portant sur les valeurs de réputations et/ou de confiance (mécanisme d'élimination, de poids ...). Nous terminerons cette section par la réalisation d'un tableau de classification récapitulatif.

2.4.1 Modèles analytiques

Les modèles analytiques sont des modèles dans lesquels les valeurs de confiance et/ou de réputation sont calculés à partir d'expressions analytique (expressions donnant toujours une valeur quelque soit les valeurs de ses paramètres). Dans cette partie nous présenterons les modèles *S.Marsh*, *Sporas et Histos*, *Regret*, *RIPSec* et nous conclurons par une exploration des modèles de réputation utilisée dans le cadre du e-commerce.

S. Marsh (1994)

L'un des tout premier modèle à proposer l'utilisation du concept de confiance fut réalisé en 1994 par [27] au sein d'une architecture multi agents. Dans son travail Marsh dérive à partir des informations de première main ce qu'il appelle l'*Expérience* lui permettant de calculer en fonction de trois types de confiance (basique, génèrale et situationnelle) la confiance qu'il attribut à un autre agent. La confiance de base est calculée à partir de l'ensemble de l'*Expérience* accumulé par un agent. La confiance situationnelle est calculée à partir de la formule 2.1 en tenant compte d'une situation (ex : un niveau de risque) alors que la confiance générale est calculée sans tenir compte de celle-ci.

$$T_x(y, \alpha) = U_x(\alpha) \times I_x(\alpha) \times \widehat{T_x(y))} \tag{2.1}$$

Dans la formule 2.1, $T_x(y, \alpha)$ détermine la confiance situationnelle accordé par un agent x à un agent y dans une situation α. $\widehat{T_x(y))}$ symbolise la confiance générale que l'agent x attribue à l'agent y et fait office de possible poids pour les évaluations antérieures de x par rapport à y. $I_x(\alpha)$ représente un poid symbolisant l'importance de la situation alors que $U_x(\alpha)$ symbolise son utilité. En plus d'une valeur de confiance pour y, le modèle va calculer un niveau de compétence requis en rapport avec la situation en cours. Si le risque est trop élevé alors le niveau de compétence requis par x augmentera suivant la formule 2.2 où T_x représente la confiance de base (valeur arbitraire) qu'attribue x à y. Notons que si le modèle doit interagir avec un agent inconnu (connaissance vierge)

il attribuera un niveau de confiance qualifié d'incertain pour celui-ci et correspondant à une valeur arbitraire.

$$PerceivedCompetence_x = T_xy \times I_x(\alpha) \tag{2.2}$$

Le modèle prendra finalement la décision de coopérer avec l'agent y en fonction d'un seuil de coopération qui sera obtenu depuis la formule 2.3 à partir de :

- $I_x(\alpha)$: l'importance de la situation α
- $\widehat{T_x(y))}$: la confiance générale de x en y,
- $PerceivedCompetence_x(y, \alpha)$: le niveau de compétence requis par x pour y dans la situation α,
- $PerceivedRisk_x(\alpha)$: le niveau de risque de la situation α perçu par x.

Le seuil de coopération obtenu sera alors comparé à la confiance situationnelle de y calculé par x, si cette dernière est supérieure alors l'agent x collaborera avec l'agent y.

$$CooperateThreshold_x(\alpha) = \frac{PerceivedRisk_x(\alpha)}{PerceivedCompetence_x(y, \alpha) + \widehat{T_x(y))}} \times I_x(\alpha) \tag{2.3}$$

Regret (2003)

Publié en 2003 par [30], Regret est un modèle de confiance et de réputation le plus utilisé dans les systèmes multi agents. Dans celui-ci l'auteur décrit la confiance comme un concept à multiples facettes et définit la réputation comme une fonction utilisant trois sources d'informations : les informations directes (1er main), les recommandations de tiers (seconde main) et les structures sociales (ontologique). Durant son exécution le système maintient à jour trois bases de connaissances représentant chacune des sources d'informations. Nous retrouvons alors trois types de réputations :

- La réputation directe calculée à partir des informations de première main.
- La réputation du voisinage calculé depuis les informations contenu dans les relations entre les différents partenaires du système.
- La réputation du système qui est calculé à partir des rôles et de propriétés générales.

En plus des trois bases de connaissance, l'agent embarque un module de crédibilité lui assurant la possibilité de mesurer un degré de fiabilité pour chaque information qu'il obtient. Le niveau du degré de fiabilité permet de déterminer si une information est bonne à prendre en compte et ainsi éviter d'enregistrer dans les bases de connaissances des informations incorrectes ou provenant d'agent non fiable. Chacune des bases de connaissances travaille de concert avec les autres pour offrir aux agents la possibilité de choisir

les réputations qu'ils souhaitent calculer en fonction de paramètres. Cette particularité du modèle permet aux agents de prendre des décisions en fonction du contexte qu'ils rencontrent et de s'adapter à différents degrés de connaissance de l'environnement, autorisant le système à être utilisé dans de large cas d'utilisation allant des plus simples au plus complexe.

Regret associe aussi à chaque valeur de confiance ou de réputation une valeur de fiabilité permettant à l'agent de ressentir à quelle hauteur le système est confiant dans la valeur calculée en fonction de la méthode qu'il a utilisée. Ce mécanisme offre à l'agent la possibilité de se situer sur la pertinence des valeurs qu'il a calculées et de décider de les inclure ou non dans son mécanisme de décision. Une autre caractéristique du modèle Regret est le fait qu'il utilise une structure ontologique. Dans leurs recherches les auteurs ont considéré la confiance et la réputation comme n'étant pas un simple concept abstrait, mais comme un concept à multiples facettes. L'utilisation de la structure ontologique permet de fournir les informations nécessaires pour combiner les valeurs de confiance et de réputation pour calculer des attributs beaucoup plus complexes. Les attributs complexes permettent de faire correspondre une réputation comme étant la combinaison de plusieurs autres réputations. Exemple : La réputation d'une compagnie de transport est la combinaison de la réputation de ses véhicules de transport (confort, ponctualité...) et de ses services aux clients (restauration, réservation ...). L'utilisation des attributs complexes permet de faire correspondre une combinaison de réputation et de poids à une attente de l'agent en termes de service.

Sporas et Histos (1999)

Sporas et Histos ([35]) sont deux systèmes de réputation en ligne complémentaire. Sporas est un simple modèle de confiance dans lequel seules les deux plus récentes évaluations entre deux utilisateurs sont considérées. Une de ses particularités majeures est que les utilisateurs ayant un haut réputation expérimentent des fluctuations plus faibles de leur niveau de réputation que les utilisateurs possédant une faible réputation. La réputation est calculée sur l'intervalle [0,1 ;1[et les nouveaux utilisateurs se voient attribuer une réputation égale à zéro. La réputation d'un agent x est mise à jour à partir des formules suivantes :

$$R_{t+1} = \frac{1}{\theta} \sum_{i=1}^{t} \Phi(R_i) R_{i+1}^{other} (W_{i+1} - E(W_{i+1})) \tag{2.4}$$

$$\Phi(R_i) = 1 - \frac{1}{1 + e^{\frac{-(R-D))}{\sigma}}} \tag{2.5}$$

Où,
R_{t+1} correspond à la nouvelle valeur de réputation
R correspond à la valeur actuel de réputation de x
t est le nombre d'évaluations fourni pour x par des recommendeurs
θ est une constante entière plus grande que 1 fixé arbitrairement
W_{i+1} représente l'évaluation donnée par le recommendeur $i+1$
R_{i+1}^{other} est la réputation du recommendeur $i+1$ fournissant l'évaluation
D est l'intervalle de la valeur de réputation (ex : si l'intervale est [0 ;1], D vaudra 1)
σ est le facteur d'accélération de la fonction Φ

Le second modèle Histos a été réalisé pour combler le manque de personnalisation de Sporas au niveau des valeurs de réputation. Le modèle Histos amène un mécanisme permettant de prendre en compte les informations dont l'agent est témoin (promiscuité) contrairement à Sporas. Ainsi la réputation est mise à jour à partir d'un agent recommendeur z et de la formule 2.6.

$$R_{t+1} = \frac{\sum_{t-\theta'}^{t} \Phi(R_{i+1})(R_{i+1}^{other} W_{i+1})}{\sum_{t-\theta'}^{t} R_{i+1}} \tag{2.6}$$

Où
$\theta' = min(\theta, m)$ et $m =$ le nombre de chemins allant de x à z.

Le reste des valeurs utilisées par Histos étant les mêmes que celles utilisées par Sporas, la réputation diminuera jusqu'à atteindre la valeur d'un nouvel utilisateur amenant une hypothèse irréaliste par le fait qu'un utilisateur ayant une très mauvaise réputation aura une réputation proche ou égale à celle d'un nouvel arrivant. Par déduction ce modèle sera incapable de distinguer un malicieux de type *Selfish* d'un nouvelle arrivant.

RIPSec (2011)

Le modèle RIPSec [22] est un modèle conçu pour apporter de la sécurité dans les réseaux mobile sans fil via l'utilisation d'un mécanisme simple pour calculer la réputation

d'un agent uniquement à partir des informations de première main. Pour calculer la réputation, le modèle parcourt les informations de première main qu'il possède sur l'agent et génère deux valeurs :

- α qui correspond aux nombres d'évaluations positives contenues dans les informations de première main.
- β qui correspond aux nombres d'évaluations négatives contenues dans les informations de première main.

La réputation de l'agent est ensuite obtenue par une simple soustraction : $DR_{ab} = \alpha_b - \beta_b$. Si la réputation directe de l'agent B calculé par A (DR_{ab}) est ≤ 0 alors l'agent A détectera l'agent B comme étant un agent malicieux et refusera toutes interactions avec lui.

Modèles de réputation en ligne de e-commerce

Dans le domaine des places de marchés en ligne eBay [2] et Amazon [1] sont parmi les plus représentatifs. Le site de vente aux enchères d'eBay utilise un mécanisme de calcul d'une valeur de réputation d'un utilisateur en se basant sur l'ensemble des évaluations fournies par les autres utilisateurs après la réalisation d'une transaction. Lors de l'évaluation, un utilisateur peut procéder à trois choix d'évaluation :

- Une évaluation positive : $+1$
- Une évaluation négative : -1
- Une évaluation neutre : 0

La réputation d'un utilisateur est calculée en additionnant l'ensemble des évaluations sur les six derniers mois. Le principe est similaire pour Amazon. Les mécanismes utilisés par ces deux sites de e-commerce ne basent leur réputation que sur les informations qui proviennent d'utilisateurs ayant déjà interagi avec l'utilisateur évalué et ne prenant pas en compte les fausses informations que pourraient fournir ces mêmes utilisateurs.

2.4.2 Modèles utilisant la logique floue

Les modèles utilisant la logique floue s'appuient sur la théorie des ensembles flous. Contrairement à la logique de Boole, cette logique permet à la valeur de vérité d'une condition de parcourir un domaine plus vaste que la simple pair ($vrai$,$faux$) offrant la possibilité de définir des degrés de satisfaction d'une condition. Ces degrés de satisfaction sont utilisés par les modèles de réputation et de confiance pour définir des intervalles d'attribution de confiance. Nous présenterons dans cette section les modèles *AFRAS* et *TRIP*.

AFRAS (2002)

Le modèle AFRAS [9] représente les valeurs de réputation grâce à des ensembles flous. Lorsqu'une interaction avec un agent prend fin, un nouvel ensemble flou est utilisé pour calculer le degré de satisfaction de l'interaction. Cette satisfaction est ensuite agrégée avec l'ancienne réputation puis combinée avec un poids pour calculer la nouvelle réputation de l'agent. Le poids utilisé pour l'agrégation est calculé à partir d'une valeur que les auteurs appellent *remembrance* ou *memory* permettant à l'agent d'attribuer plus d'importance (poids) à la dernière interaction ou aux valeurs de réputation anciennes. Cette valeur *memory* est représenté sous la forme d'une fonction de similarité entre :

- La précédente valeur de réputation et la valeur de satisfaction (poids) de la dernière interaction.
- La précédente valeur de la valeur *memory*.

Ainsi si la précédente valeur de satisfaction et l'ancienne valeur de réputation sont similaires alors la pertinence des informations de première main est augmentée, au contraire si les deux valeurs sont différentes alors c'est seulement la pertinence de la dernière interaction qui sera augmentée. Les informations provenant d'autres agents (informations de seconde main) sont agrégées avec les informations de première main en fonction d'un poids dépendant de la réputation du recommendeur (agent auprès duquel nous obtenons des informations de seconde main sur notre cible). Plus la réputation du recommendeur sera élevée et plus les informations de seconde main obtenues seront considérées comme des informations de première main, à l'inverse si la réputation de celui-ci est faible (seuil arbitraire) les informations qu'il offrira seront tout simplement non prises en compte.

TRIP (2012)

Le modèle Trip [16] est un modèle conçut pour être utilisé dans un environnement MANET et plus particulièrement dans un environnement VANET (Vehicule Ad-hoc NETwork) qui fait correspondre des véhicules réalisant le rôle d'agent du système. Dans celui-ci les véhicules fournissent des informations concernant le réseau routier comme la présence d'obstacles, les embouteillages ou des difficultés causées par la météo. Durant leur parcours les véhicules peuvent rencontrer d'autres véhicules ou ce que le modèle appelle des *Road Side Unit* (RSU) qui sont des antennes locales partageant des informations de réputation sur certains véhicules du réseau. Ainsi le modèle calcule la réputation d'un véhicule à partir de 3 sources d'informations :

- La première source correspondant aux informations de première main.

– La seconde source consistante aux informations de seconde main fournit par des recommandeurs

– La troisième source étant les informations de réputation obtenue auprès des RSU.

Ces trois sources sont ensuite agrégées suivant la formule 2.7 pour obtenir la réputation globale d'un véhicule.

$$OR_{ab} = \alpha_a DR_{ab} + \beta_a \sum_{i=1}^{n} \omega_i IR_{ib} + \gamma_a DR_{RSUb} \tag{2.7}$$

Avec

$\alpha_a + \beta_a + \gamma_a = 1$ et qui correspondent au poids que l'agent a attribue à chaque source d'informations

ω_i qui représente la fiabilité de l'agent i

DR_{RSUb} correspondant à la réputation directe de l'agent b calculé à partir des informations obtenues auprès des RSU.

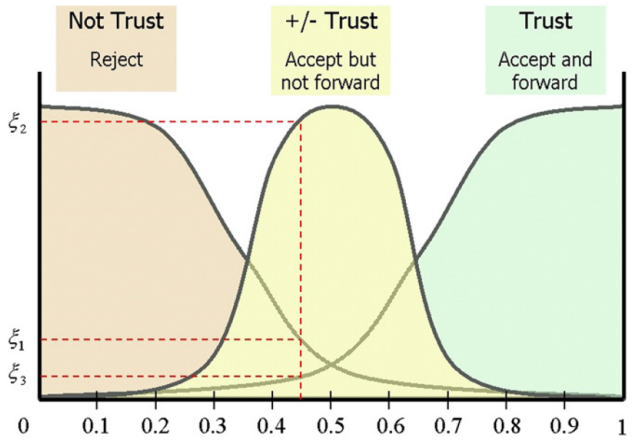

FIGURE 2.10: Niveau de confiance dans Trip (Source : [16])

Ensuite la réputation globale d'un agent est injectée dans un ensemble flou offrant trois niveaux de satisfaction. En fonction du résultat, un niveau de confiance (Figure 2.10) est attribué et une décision est prise pour l'information fournie par l'agent :

– **Non digne de confiance** : Rejet de l'information.

– **Plus ou moins digne de confiance** : Aggrégation de l'information, mais pas de transfert de celle-ci à d'autres agents.

– **Digne de confiance** : Aggrégation et relais de l'information aux agents voisins.

2.4.3 Modèles imitant des comportements biologiques

Cette section regroupe les modèles de réputation et de confiance qui calquent leur comportement par rapport à des comportements biologiques existants. Nous présenterons dans cette section les modèles *AntRep* et *QDV* s'inspirant tout les deux des comportements biologiques des Fourmis.

AntRep (2006)

AntRep [33] est un modèle utilisé dans le cadre d'un réseau P2P ou MAS et prévu dans le cadre d'échange de message entre agents. Dans celui-ci chaque participant possède une table de réputation (RT) qui exerce une fonction similaire aux tables de routage sauf que la valeur de réputation stockée sert à indiquer la probabilité de choisir un voisin comme étant le prochain saut. AntRep utilise l'envoi de fourmis sur le réseau suivant deux cas :

- Le modèle enverra des fourmis de type *Unicast* au voisin possédant la plus haute réputation par rapport au reste de son entourage.
- Le modèle enverra des fourmis de type *Broadcast* à tout son entourage s'il ne possède aucune information sur son entourage.

Les fourmis envoyées joueront un rôle d'éclaireur pour le modèle et continueront à avancer dans le réseau jusqu'à trouver ce qu'elles recherchent (un fournisseur de service ou un destinataire). Une fois leur candidat trouvé elles gêneront une fourmi de type *Backward* dont le rôle sera de revenir jusqu'au modèle en faisant chemin inverse. Durant leur retour elles mettront à jour une valeur de réputation à chaque noeud i traversé en respectant la formule 2.8.

$$P_i(t) = \frac{[\tau_i(t)]^\alpha [\eta_i(t)]^\beta}{\sum_{j \in N} [\tau_j(t)]^\alpha [\eta_j(t)]^\beta} \tag{2.8}$$

Avec η_i la valeur de réputation attribué par le noeud i avec son voisin j.
τ_i la phéromone définie comme étant au temps $t+\Delta t$ par un noeud j recevant une fourmi de type *Backward* par le noeud i :

$$\tau_i(t + \Delta t) = f(\tau_i(t), \Delta t) + \Delta p \tag{2.9}$$

$$\tau_j(t + \Delta t) = f(\tau_j(t), \Delta t), j \in N, j \neq i \tag{2.10}$$

Avec $\Delta p = \frac{k}{f(c)}$, $k > 0$ et $f(c)$ non décroissant et c étant un paramètre (arbitraire) du scénario du réseau.
$f(\tau_i(t), \Delta t)$ correspondant à la fonction d'évaporation de la phéromone :
$f(\tau_i(t), \Delta t) = \frac{\tau_i(t)}{e^{\frac{\Delta t}{k}}}$ et α et β étant des constantes variant suivant l'environnement du réseau.

La phéromone est aussi utilisée pour décider quand envoyer une fourmies *Broadcast* sur le réseau, ce choix à lieu lorsqu'un noeud reçoit une requête pour laquelle il n'a pas encore d'entrée dans sa *RT*. S'il en possède une, il choisira le voisin possédant la plus haute réputation comme prochain saut.

QDV (2009)

Dans QDV [12], les auteurs proposent une approche de qualité de service basé sur la sécurité dans les réseaux de senseur sans fil. Ils utilisent un concept similaire à *AntRep* au niveau de la table de réputation se basant sur le niveau de réputation pour déterminer si une sonde est fiable pour la communication. Le modèle utilise la promiscuité présente dans les réseaux sans fil pour observer le rejet par les sondes voisines des paquets transférés. Ce mécanisme a pour but d'identifier les sondes malicieuses afin d'augmenter la sécurité du réseau et plus directement la communication entre les sondes. QDV se base sur les chemins de communication (routes) et les classifie selon le nombre de phéromones (τ_i) présentes (équivalent à des recommandations). Ainsi le chemin le plus marqué par des phéromones est déclaré comme étant le plus sûr. Néanmoins le modèle prend aussi en compte la distance (en terme de saut) entre deux sondes (η_{ij}) dans le calcul de la réputation suivant la formule 2.11 :

$$\Phi_{ij}(t) = \frac{\sum_{k=1}^{n_i} \tau_{kj}}{\eta_i} \tag{2.11}$$

Où τ_{kj} représente la trace de phéromone entre la sonde k et la sonde j
η_i correspond aux nombres de voisins de la sonde i.

Ainsi la détection d'un mauvais comportement ou d'une violation de sécurité se fait si $\Phi_{ij}(t) < \tau_{min}$. Le modèle représente aussi la qualité de service comme étant le pourcentage de trafic exposé suivant la formule 2.12.

$$\theta_{ij}(t) = \frac{\sum M_g(t) + \sum M_r(t) + \sum M_d(t)}{\sum M_g(t) + \sum M_r(t)} \eta_{ij} \tag{2.12}$$

Avec $\sum M_g(t), \sum M_r(t), \sum M_d(t)$ représentant dans l'ordre le nombre de paquets générés, reçus et rejetés. Un dernier concept appelle par le modèle la qualité de sécurité (QSec) dépendant des formules 2.11 et 2.12 définit la communication entre deux sondes. La QSec est un facteur de décision (niveau de confiance) grâce auquel une sonde peut-être sélectionné comme étant le prochain saut du chemin et est calculée en réalisant la somme pondéré de la réputation et de la qualité de service suivant la formule 2.13.

$$W_n(t) = w_1\Phi_{ij}(t) + w_2\theta_{ij}(t) \tag{2.13}$$

Où w_1 et w_2 sont les poids respectifs attribués à la réputation et à la qualité de service et $w_1 + w_2 = 1$.

2.4.4 Modèles utilisant les réseaux Bayésiens

Dans cette section nous présentons les modèles basant leur calcules sur la théorie des Réseaux Bayésien et plus particulièrement l'utilisation de la fonction de densité de probabilité provenant de la Loi bêta pour le calcul des valeurs de réputation (formule 2.14).

$$B[\alpha, \beta] = \frac{\alpha}{\alpha + \beta} \tag{2.14}$$

Nous présenterons les modèles *Confidant*, *RFSN*, *Travos* et *Arman* en détaillant les valeurs qu'ils attribuent à α et β lors du calcule d'une valeur réputation et/ou d'un niveau de confiance.

Confidant (2004)

Le modèle de réputation Confidant [8] est un modèle français conçu pour fonctionner dans des environnements de type MANET et P2P. La réputation d'un agent B calculé par un agent A est obtenue grâce à une seule méthode comprenant deux phases. La première phase consiste à calculer la réputation de l'agent à partir des informations de première main, pour ce faire les informations de première main sont parcourues une par une. Lors du parcours les valeurs de α et β sont initialisées à 0 et incrémentées suivant la formule 2.15 si l'évaluation est positive (augmentation de la valeur de α) et selon la formule 2.16 si l'évaluation est négative (augmentation de la valeur de β).

$$\forall i \in n : \alpha_i = \alpha_{i-1} \times u + 1 \tag{2.15}$$

$$\forall j \in m : \beta_j = \beta_{j-1} \times u + 1 \tag{2.16}$$

Avec u une constante valant $0,999$.
n le nombre d'évaluation positive.
m le nombre d'évaluation négative.

Une fois le parcours des informations de première main terminée, les valeurs de α et β sont injectées dans la fonction de densité Beta (2.14) pour obtenir la réputation directe de l'agent (DR). La seconde phase va interroger l'ensemble des agents voisins (sauf l'agent pour lequel nous calculons sa réputation) pour obtenir leurs évaluations directes. Ces évaluations indirectes obtenues auprès de recommendeurs seront parcourues pour chacun des recommendeurs comme en phase 1 permettant d'obtenir les réputations indirectes (IR) que les recommendeurs fournissent à propos de l'agent que nous évaluons. Pour terminer la réputation directe et additionner aux réputations indirectes qui réussissent un test de déviation et de confiance suivant la formule 2.17 pour obtenir la réputation globale (OR) de l'agent.

$$OR_{ab} = DR_{ab} + \sum_{i}^{n} w \times IR_{ib} \tag{2.17}$$

Avec
a, b, i des agents
n le nombre de voisins de a ayant réussi le test de déviation et de confiance
w une constante égale à $0,1$

Le test de confiance consiste à vérifier que la réputation directe du recommendeur C (DR_{ac}) est $\geq 0,25$, tandis que le test de déviation compare, au fur et à mesure de la somme des réputations indirectes, la valeur actuelle de la réputation globale (OR_{ab}) avec la prochaine réputation indirecte qui sera ajoutée (IR_{ib}). Si celles-ci diffèrent de plus $0,5$ alors la prochaine réputation indirecte est écartée du calcul de la réputation globale.

RFSN (2008)

RFSN [15] est un modèle pensé pour les réseaux de senseurs sans fil dans lequel la réputation d'un agent B par un agent A est calculée comme dans le modèle *Confidant* en deux phases. La première phase se contente d'utiliser la même approche que *Confidant* en utilisant la formule 2.18 pour α et 2.19 pour β. Sauf qu'une fois les valeurs de α et β obtenu celles-ci ne sont pas injecté tout de suite dans le fonction de densité Beta pour obtenir une valeur de réputation.

$$\forall i \in n : \alpha_i = (w \times \alpha_{i-1}) + i \tag{2.18}$$

$$\forall i \in j : \beta_j = (w \times \alpha_{j-1}) + j \tag{2.19}$$

Avec
w une constante de vieillissement.
n le nombre d'évaluation positive.
m le nombre d'évaluation négative.

La seconde phase va s'adresser aux recommendeur et obtenir leur évaluation indirecte. Pour chacun des recommendeurs interrogés, la même méthode qu'en phase 1 sera appliquée pour générer un couple (α_{IR}, β_{IR}). Ces couples (α_{IR}, β_{IR}) issuent des informations de seconde main seront agrégés avec le couple (α, β) provenant de la phase 1 suivant les formules 2.20 et 2.21 pour obtenir un dernier couple (α_{OR}, β_{OR}) qui sera injecté dans la fonction de densité Beta pour ainsi obtenir la réputation globale de l'agent évalué. La réputation globale ainsi obtenue permettra d'atteindre le niveau de confiance de l'agent.

$$\alpha_{OR} = \sum_i \frac{2 \times \alpha_{IR_i} \times \alpha}{(\beta_{IR_i} + 2) \times (\alpha + \beta + 2) + (2 \times \alpha_{IR_i})} \tag{2.20}$$

$$\beta_{OR} = \sum_i \frac{2 \times \alpha_{IR_i} \times \beta}{(\beta_{IR_i} + 2) \times (\alpha + \beta + 2) + (2 \times \alpha_{IR_i})} \tag{2.21}$$

Travos (2006)

Travos [31] est un modèle développé pour les systèmes multi agents dans lequel un agent utilise deux méthodes pour mesurer la fiabilité d'autres agents du système en se basant sur les informations de première et seconde main. La première méthode est basée

sur la fonction de densité Beta et les informations de première main. Lorsque la fiabilité est calculée avec la première méthode, un niveau de conviction est défini pour déterminer si l'agent est confiant sur la suffisance de ses informations de première main. Si le niveau de conviction est trop bas alors l'agent va avoir recours à la seconde méthode consistant à utiliser les informations de seconde main pour affiner son jugement. Lors de la collecte par le modèle des informations de seconde main, celles-ci sont graduellement comparées avec la majorité (constitué au fur et à mesure de la durée de vie de l'agent) et pondérées avec un poids. Plus les informations d'un recommandeur seront éloignées de l'opinion de la majorité plus le poids attribué à ces informations sera faible. Une fois toutes les informations de seconde main obtenue et pondérée, celles-ci sont agrégées pour obtenir la réputation de l'agent. Le mécanisme utilisé par Travos tente de confondre, lors de l'utilisation de la seconde méthode, l'opinion de la majorité aux opinions individuelles et de diminuer l'importance des opinions individuelles divergentes en leur attribuant une pondération faible lors de l'agrégation de toutes les opinions. Cette approche se base sur l'hypothèse que toutes les informations fournies par les recommendeurs sont cohérentes sans prendre en compte que les agents peuvent tricher et fournir de fausses informations.

Arman (2013)

Le modèle Arman [18] est un modèle pensé pour les environnements de type MANET dans lequel la réputation globale (OR) d'un agent B est calculée par un agent A grâce à deux sources d'informations. La première est constituée des informations de première main et permet le calcul de la réputation directe (DR) de B selon A. La seconde source d'information est liée aux informations de seconde main obtenue auprès des recommandeurs et pondérée par une valeur de fiabilité (λ), la valeur obtenue représentant la réputation indirecte de l'agent B (IR). Les valeurs des réputations directes et indirectes sont calculées à partir de la fonction de densité Beta en injectant α et β qui correspondent respectivement au nombre d'évaluations positives et négatives. Ces deux valeurs sont ensuite agrégées selon la formule 2.22.

$$OR_{ab} = \delta DR_{ab} + \frac{(1-\delta)\sum_{i=1}^{n} \lambda_{ai} IR_{ib}}{\sum_{i=1}^{n} \lambda_{ai}} \tag{2.22}$$

La fiabilité λ est calculée comme étant un indice correspondant à la similitude de vue entre l'agent réalisant l'évaluation et le recommandeur fournissant les évaluations. Le principe de la similitude de vue est de se dire que si deux agents observent les mêmes comportements alors leurs opinions doivent être les similaires. Ainsi en prenant l'agent A et le recommandeur C, l'agent A va recenser l'ensemble des agents pour lesquels lui

et le recommandeur C ont réalisé une évaluation ($Set(A, C)$). Cet ensemble d'agents est alors utilisé pour calculer la similitude de vue selon les formules 2.23 et 2.24.

$$sim(A, C) = 1 - dis(A, C) \tag{2.23}$$

$$dis(A, C) = \sqrt{\sum_{i \in Set(A,C)} (DR_{Ai} - DR_{Ci})^2} \tag{2.24}$$

Lors du calcul de la fiabilité en utilisant la similitude de vue le résultat de la différence ($DR_{Ai} - DR_{Ci}$) est comparé à un seuil. Si ce seuil est dépassé, le recommendeur est automatiquement considéré comme étant malicieux et est écarté par le modèle.

Une fois la réputation globale obtenue, un dernier test est réalisé pour obtenir un niveau de confiance en comparant la réputation globale calculée par l'agent aux recommandations du voisinage grâce à la loi de combinaison de *Dempster-Shafer* (aussi appelé somme orthogonale). Si cette dernière est supérieure à la combinaison selon *Dempster-Shafer* des recommandations et si celle-ci est aussi supérieure à un seuil (égale à la réputation par défaut attribué par le modèle), alors le niveau de confiance est suffisant pour accepter la transaction de l'agent. Dans le cas contraire, celle-ci sera simplement rejetée.

2.4.5 Tableau de classification des modèles

Dans cette section nous présentons un tableau récapitulatif des modèles (tableau 2.1) que nous avons présenté aux points précédant. Ceux-ci se retrouvent classés par environnement et méthodologie propre à leurs fonctionnements. Pour réaliser ce tableau nous nous basons sur les classifications de [26] et [30] ainsi que sur les lectures relatives à chacun des modèles présentés.

2.5 Métriques de comparaison de modèles

Pour comparer les modèles entre eux, de nombreux auteurs ont recours à des métriques collectées durant les simulations des modèles. Provenant soit d'un agent en particulier ou reflétant le système tout entier, elles peuvent être collectées en fin de simulation ou observées durant son déroulement. Dans son article, [11] analyse 31 articles utilisant des métriques et établit pour chacune d'elles un taux d'utilisation au sein des articles analysés. En tête de liste nous retrouvons les métriques symbolisant la capacité de détection des comportements malicieux un modèle suivit par des métriques relatives à la consom-

	Analytique	Logique Floue	Biologique	Bayésien
MAS	Sporas & Histos Regret S.Marsh eBay/Amazon	AFRAS Venanzi	AntRep	
P2P			AntRep	Confidant
WSN			QDV	RFSN
MANET	RIPSec	Trip		Confidant Arman

TABLE 2.1: Tableau récapitulatif des modèles présentés

mation du modèle dans le réseau (nombre de paquets émits, taux d'usage d'une route ...). D'autres articles comme [25] et [29] ajoute de nouvelle métrique comme la précision d'un modèle (taux d'erreur dans le choix des candidats potentiels) ou encore le temps utilisé par le modèle pour calculer une valeur de réputation ou de confiance. Nous tentons dans le tableau 2.2 de classer les différentes métriques reprises dans ces articles en les classant sous plusieurs axes :

- La portée de la métrique qui est propre à un agent ou représentative du système entier.
- L'évaluation de la métrique représentant une valeur liée au modèle de réputation et/ou de confiance ou une valeur liée au réseau.

	Agent	Système	Description
Modèle	Valeur de Réputation		Évolution de la valeur de réputation calculée pour un agent au fil du temps.
	Niveau de confiance		Évolution du niveau de confiance calculé pour un agent au fil du temps.
	Fiabilité / poids		Évolution des poids/valeur de fiabilités calculés pour un agent au fil du temps.
	Délais de calcul	Moyenne de délais de calcul	Temps réalisé par le modèle pour calculer une valeur de réputation ou un niveau de confiance (récupération des évaluations et agrégations).
	Taux de détection	Taux de détection globale	Pourcentage d'agent malicieux présent dans le réseau et détecté.
	Nombre de détection	Nombre total de détections	Nombre d'agents détecté comme étant malicieux au cours d'une simulation.
	Temps de détection	Moyenne du temps de détection	Temps moyen (en tour ou en seconde) nécessaire à un agent pour considérer un autre agent comme malicieux.
	Taux de précision	Moyenne du taux de précision	Ratio entre le nombre de services demandés fournissant un niveau de satisfaction suffisant et ceux fournissant un niveau de satisfaction insuffisant.
		Taux d'agent malicieux	Pourcentage d'agent malicieux présent dans le réseau.
	Consommation d'énergie	Consommation totale d'énergie	Valeur énergétique consommée par le modèle (calculé sur base de son temps de calcul, le nombre d'évaluations téléchargé auprès de recommandeurs...).
		Taux d'utilisation d'une route	Nombre de fois où une route est empruntée.
Réseau	Paquet émit	Total des paquets émis	Nombre total de paquets émis par un agent ou par le réseau entier.
	Paquet perdu	Total des paquets perdus	Nombre total de paquets ayant été rejeté par le réseau.
	Nombre de service	Total du nombre de services	Nombre total de services demandé ou presté par un agent ou le réseau entier.

TABLE 2.2: Classement des métriques de comparaison de modèles

2.6 Simulateurs d'environnement pour modèles

Il existe une pléthore de simulateur pour chacun des environnements que nous avons présentés en 2.2.2 notamment [32] et [19] pour les systèmes multi agents. GlomoSim [5] et OMNET++ [21] pour les MANETs. Ainsi que beaucoup d'autres pour les P2P [28] et WSN [34].

Néanmoins très peu d'entre eux proposent de simuler les modèles de réputation et/ou de confiance au sein de leur environnement. Nous présenterons dans cette section trois simulateurs de modèles de réputation et/ou de confiance spécifique à un environnement et détaillerons pour chacun d'entre eux les comportements malicieux et les différentes métriques qu'ils proposent. Ainsi nous présenterons *ART Testbed* qui est l'un des simulateurs les plus utilisés dans les environnements multi agent, *TOSim* pour les environnements P2P et *TRMSim-WSN* qui est l'unique simulateur de modèles en environnement WSN existant. Nous ne présenterons pas de simulateurs de modèles pour les MANETs car ceux-ci sont actuellement inexistants.

2.6.1 TOSim (P2P)

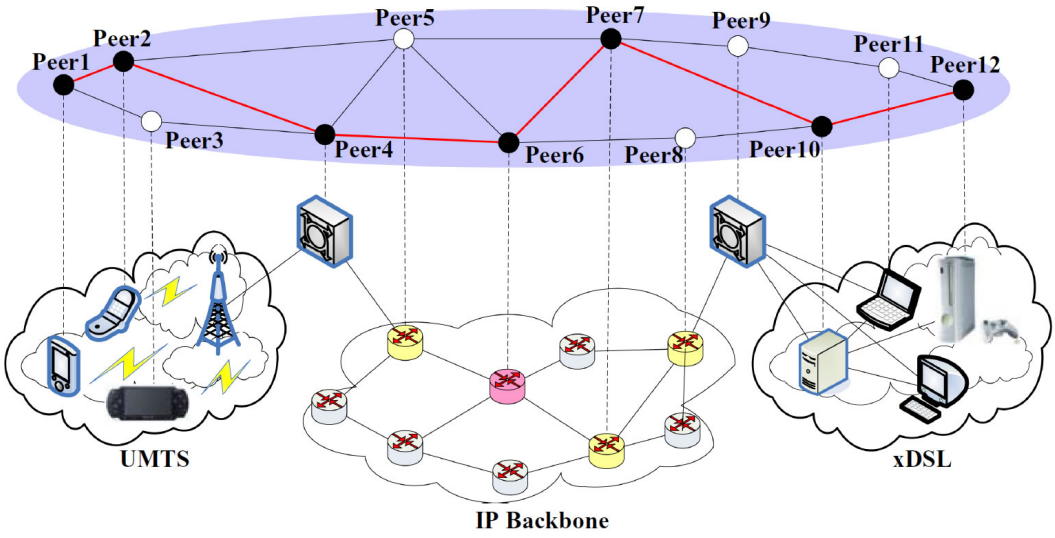

FIGURE 2.11: Architecture d'un réseau général avec TOSim (Source [36])

TOSim [36] est un simulateur d'environnement *P2P* pour modèle de réputation et/ou de confiance optimisé pour simuler des réseaux *P2P* de grande taille (plusieurs millions de pairs) avec un minimum de consommation de mémoire et de temps de réaction. Le

simulateur simule des réseaux composés de pairs pouvant supporter un ou plusieurs protocoles de communication (Figure 2.11) tout en proposant la gestion en temps réel d'apparition et de disparition de pairs dans le réseau (réseaux ad-hoc). Le simulateur TOSim propose entre autres de confronter les modèles aux malicieux de types *Oscillants* et *Ballot-Stuffing* dans un scénario de test décentralisé. A cet effet le protocole de test se présente sous deux phases complémentaires. La première phase est une phase de recherche de pair parmi le voisinage susceptible de fournir un fichier recherché par l'agent (recherche d'un fournisseur de service). Une fois ce pair découvert le fichier est téléchargé auprès de la source (réalisation du service) pour ensuite être testé sur son intégrité par l'agent (évaluation du service). TOSim propose uniquement la simulation de réseau filaire. Le simulateur permet aux utilisateurs de consulter l'évolution des valeurs de confiance calculé par le réseau (moyenne) au fil du temps comme métrique de comparaison pour les modèles ainsi que consulter des informations statistiques sur le réseau (nombre de messages échangés, nombre de fichiers échangés ...). TOSim propose l'incorporation de nouveau modèle sous la forme d'algorithme de décision au sein de ses agents. Le moteur de simulation de TOSim est un moteur de type *time-step* où durant un nombre de tours définit chaque agent du réseau sera exécuté exactement une fois dans un ordre aléatoire.

2.6.2 ART (MAS)

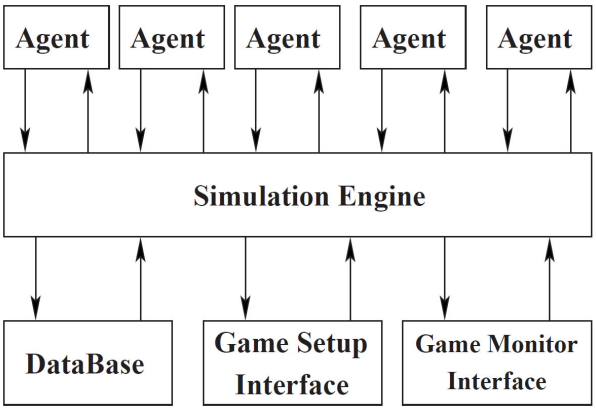

FIGURE 2.12: Architecture de ART Testbed (Source [13])

ART Testbed (Agent Reputation and Trust) [13] est un simulateur modulaire permettant de tester des modèles de réputation et/ou de confiance dans le cadre d'un environnement multi agent et proposant une paramétrisation poussée de celui-ci. Le

simulateur ART a été modélisé (Figure 2.12) pour permettre une grande accessibilité dans l'intégration des algorithmes des différents modèles, offrant la possibilité aux utilisateurs de s'approprier le simulateur pour leurs propres expérimentations. Le protocole de ART se base sur un mécanisme de client/serveur décentralisé opposant les agents à des malicieux de type *Oscillant* (notons qu'un agent du système réalise les deux rôles) dans lequel un premier agent jouant le rôle de client requerra auprès d'un second agent réalisant un rôle de serveur la réalisation d'un service. Le service rendu par le serveur est un service de génération d'opinion dépendant de son expertise. Les auteurs illustrent cette génération d'opinion dans leur simulateur en distinguant les serveurs comme étant des experts en évaluation de peintures dans un ou plusieurs domaines (catégorie de service). Ainsi lorsqu'un client demande l'évaluation payante d'une peinture à un serveur, celui-ci va d'abord fournir au client une valeur représentant son degré d'expertise dans le domaine de la peinture soumise par le client (il peut mentir) ainsi qu'un prix non négociable. Si le client est satisfait du degré d'expertise du serveur, il payera l'expertise de sa peinture et la recevra par le serveur. Les modèles de réputation et/ou de confiance peuvent être intégrés au sein du simulateur en fonction des choix de l'utilisateur par exemple auprès des serveurs pour instaurer une stratégie de maximisation des profits (fournir les meilleures expertises dans tous les domaines, même ceux dans lesquels le serveur n'est pas expert) ou du client pour optimiser l'expertise des peintures à moindre coût (s'adresser au meilleur expert le plus rapidement possible), ou encore les deux à la fois. Entre autres le simulateur fournit pour comparer les modèles implémentés plusieurs métriques locales (propre à un agent du système) comme le nombre de messages émit ou le solde de son compte en banque, ainsi que globale au système comme le nombre total de messages émit par type, le nombre total de transactions réalisé ou encore la répartition des richesses. ART propose un moteur de simulation basé sur la même architecture de TOSim, mais avec la possibilité supplémentaire d'étendre les métriques des agents et du système.

2.6.3 TRMSim-WSN (WSN)

TRMSim (Trust and Reputation Models Simulator) [24] est un simulateur conçu pour simuler des senseurs dans un environnement sans fil sans promiscuité. Le simulateur propose de confronter les modèles de réputation et/ou de confiance à des comportements malicieux de type *Oscillants* et *Bad-Mouthing* (appelé dans l'article collusif). Le mécanisme utilisé simule des transactions sous forme de service entre les différents senseurs du réseau. Dans celui-ci lors de l'initialisation, chaque senseur se voit attribué un ou des services qu'il peut fournir ainsi qu'un ou plusieurs services dont il est demandeur.

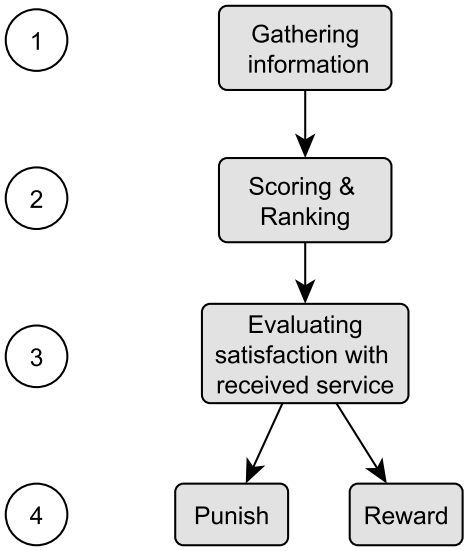

FIGURE 2.13: Protocole du simulateur TRMSim-WSN (Source [24])

Une fois tous les agents paramétrés, une topologie aléatoire de réseau est réalisée et les différents senseurs qui le composent sont connectés entre eux en fonction d'une portée d'émission maximum. Ensuite chacun d'entre eux va exécuter un protocole en 4 phases (Figure 2.13) réalisé par un moteur de simulation de type *time-step* :

1. La première étant une phase de collecte de donnée où le senseur va par le biais du simulateur obtenir la liste des senseurs voisins pouvant fournir un ou plusieurs des services dont il est demandeur.
2. La seconde phase consiste à calculer à l'aide du modèle de réputation et/ou de confiance une valeur de confiance/réputation pour chacun des fournisseurs de service à proximité. Cette phase permet de détecter les bons et les mauvais fournisseurs et de faire une sélection.
3. La troisième phase voit les transactions avec les fournisseurs de service sélectionné durant la phase 2 se dérouler. À la fin de chacune des transactions, le senseur va produire un niveau de satisfaction.
4. La dernière phase analyse les niveaux de satisfaction qu'a produite le senseur pour chacune des transactions qu'il a réalisées. Si le niveau est élevé alors le senseur va récompenser le fournisseur avec une bonne évaluation. Au contraire si le niveau de satisfaction est bas, il choisira de punir le fournisseur par une mauvaise évaluation.

Une fois que chacun des senseurs a terminé les 4 phases, une nouvelle topologie de réseau est générée aléatoirement et ainsi de suite jusqu'à l'arrêt du simulateur. Le simulateur propose trois métriques du système pour comparer les modèles qui sont :

- La distance totale parcourue par tous les messages du système.
- La précision du modèle qui représente le ratio entre le nombre total de transactions et le nombre total de transactions qui ont généré une haute satisfaction durant 1 round.
- La consommation totale d'énergie qui est une métrique propre à chaque modèle qui attribue une valeur de consommation aux opérations réalisées par le modèle s'exécuter.

TRM-Sim propose l'extension de modèle de réputation et/ou de confiance par l'intégration à celui-ci de nouveaux senseurs embarquant un modèle. Le simulateur propose aussi la paramétrisation des senseurs pour simuler différents composants de ceux-ci par exemple un niveau de batterie qui diminuerait en fonction des actions réalisées par l'agent.

2.6.4 Tableau comparatif

Nous proposons au tableau 2.3 une reprise des différentes caractéristiques des simulateurs présentés aux sections précédentes pour comparaison.

2.7 Synthèse

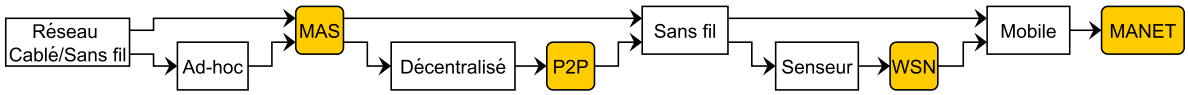

FIGURE 2.14: Hiérarchie des environnements d'utilisation des concepts de Confiance et de Réputation

La diversité des modèles de Confiance et de Réputation montrés en section 2.4, associés aux différents environnements d'utilisation (section 2.2.2) fait exploser les possibilités de choix de modèle disponible pour les concepteurs des réseaux de demain. Pour aider les concepteurs dans leurs choix, la communauté scientifique a mis au point divers simulateurs permettant d'exécuter différents modèles au sein de simulation propre à des environnements types et de les comparer. Nous avons présenté trois d'entres eux en section 2.6 permettant de couvrir les réseaux de type multi agent, pair-à-pair et de senseurs sans fil. Néanmoins, nous constatons un manquement de cette couverture pour les nou-

	TOSIM	**ART**	**TRMSim**
Fonctionnalités			
Environnement	Ad-hoc P2P	Ad-hoc MAS	Ad-hoc WSN
Architecture	Centralisé	Centralisé	Centralisé
Protocole	Recherche et échange de fichier entre pairs	Echange d'opinion d'expertise d'oeuvre d'art contre paiement	Réalisation de service entre senseurs
Métriques	Evolution des valeurs de confiances	Bénéfices, Répartition des richesses, nombre de message émit (classement par type)	Précision du modèle, Distance moyenne totale parcourue par les message, consommation d'énergie du modèle
Comportement malicieux	Oscillant, BS	Oscillant	Oscillant, BS
Topologie réseau	Dynamique (disparition/apparition d'agent)		
Type de simulation	Cycle temporelle		
Interfaces graphique	Aucune	Paramètrisation de la simulation uniquement	Paramètrisation et éxecution de la simulation
Extensions			
Protocole	Non	Non	Non
Modèle	Oui	Oui	Oui
Métrique	Non	Oui	Non
Comportements malicieux	Non	Non	Non
Informations			
Statut	N/A	Abandonné depuis 2008	Dernière MAJ : 5 avril 2013
License	N/A	LGPL 3.0	LGPL 2.0

Table 2.3: Tableau comparatif des simulateurs

veaux réseaux de type *MANETs*. Ces réseaux issus des nouveaux besoins en termes de services orientés communication ajoutent aux réseaux sans fil Ad-hoc existant l'aspect mobilité et par la sorte une topologie en constante modification.

La figure 2.14 nous montre la hiérarchie des différents environnements, ainsi que l'ensemble des aspects à prendre en compte pour dériver un environnement vers un autre. Partant de celles-ci nous constatons qu'un système multi agent est un réseau câblé/sans-fil centralisé pouvant être *Ad-hoc* (changement dynamique de la topologie par l'apparition/disparition d'agent), alors que les réseaux *P2P* sont des extensions des configurations possibles des *MAS* en décentralisé. La figure nous renseigne ensuite sur le fait que les réseaux de senseurs sans-fil sont aussi une extension à base de senseurs des configurations possibles de réseaux de type *P2P* et *MAS* tandis que les MANETs couvrent l'ensemble des configurations possibles de ces réseaux en leur rajoutant un aspect de mobilité.

Repartant de la figure 2.14 et du tableau comparatif 2.3 il devient possible d'analyser les simulateurs *ART*, *TOSim* et *TRM-Sim* afin de déterminer l'ensemble des modifications qui devront leur être apportés pour permettre la simulation des réseaux *MANETs*.

Le simulateur *ART*, permettant de simuler des modèles de réputation et/ou de confiance au sein d'environnement de type multi agent, est le seul simulateur offrant la possibilité d'étendre ses métriques. Cependant, cette particularité n'est pas suffisante pour permettre la simulation de réseaux *MANETs*. Le moteur de simulation devra être étendu pour permettre la gestion de l'aspect mobilité (déplacement, vitesse, ...) et sans-fil (promiscuité,...). Les agents devront être modifié pour permettre la prise en compte d'aspect propre aux senseurs (ex : niveau de batterie, portée...) et permettre aussi l'intégration de nouveaux comportements malicieux. Le protocole de compétition utilisé est un protocole de type client-serveur convenant à des simulations de réseaux décentralisés (chaque agent joue pour son propre compte), mais ne convient pas pour réaliser des simulations de réseaux centralisés (l'ensemble des agents coopèrent pour atteindre un but global, généralement faire parvenir l'information à un point central du réseau), de ce fait un protocole pour les scénarios centralisé devra être implémenté. *TOSim* souffre des mêmes besoins de modification que *ART* avec en plus la nécessité d'étendre les métriques. *TRM-Sim*, permettant une simulation des réseaux de senseurs sans fil, peut paraître à premier vue comme étant le simulateur requérant le moins de modifications en référence à la figure 2.14 nous montrant que l'évolution des réseaux *WSN* en *MANET* se fait au travers de l'ajout de la gestion des aspects de mobilité. La prise en compte ou non des aspects lié aux senseurs se limitant bien souvent à permettre aux agents de la simulation d'adapter leur niveau de collaboration avec le réseau en fonction

d'un niveau de batterie. Néanmoins, le simulateur *TRM-Sim* implémente de manière imparfaite les aspects liés aux réseaux sans fil par l'omission de la promiscuité. De plus celui-ci, similairement à *ART* et *TOSim*, ne propose qu'un unique protocole de type client/serveur satisfaisant des scénarios décentralisés et souffre des mêmes manquements que *TOSim* en termes de possibilités d'extensions des métriques et des comportements malicieux.

Dès lors nous constatons que les trois simulateurs, s'ils diffèrent par leur métrique implémentée et certaines spécificités provenant des réseaux qu'ils s'orientent à simuler, souffrent des mêmes manquements en termes d'aspects et de fonctionnalités pour permettre la simulation de l'ensemble des configurations possibles de réseaux *MANETs*. On retrouve parmi ces modifications communes :

- La refonte des moteurs de simulation pour intégrer les aspects de mobilité et des réseaux sans fil.
- L'intégration d'un protocole permettant la simulation de scénarios centralisés en plus d'un protocole permettant la simulation de scénarios décentralisés.
- La modification du simulateur pour permettre l'extension des métriques et des comportements malicieux.
- La possibilité de configurer l'ensemble des nouveaux paramètres de simulation

Aux vues des modifications lourdes devant être apportés aux simulateurs des modèles de réputation et/ou de confiance pour permettre la simulation de ces modèles dans les différents environnements *MANETs* existants, nous proposons la réalisation d'une nouvelle plate-forme de simulation baptisée *MANET-Sim* que nous présentons au chapitre suivant comme étant un simulateur et comparateur de modèle de réputation et/ou de confiance couvrant les environnements *MANETs*, *P2P*, *WSN* et *MAS*.

Chapitre 3

MANET-Sim

FIGURE 3.1: Logo de l'application MANET-Sim

Dans ce chapitre nous présentons notre solution *MANET-Sim* permettant de simuler et de comparer des modèles de Réputation et/ou de Confiance (MRC) au sein des différents environnements d'utilisations que nous avons observées en 2.2.2 et détaillons le fonctionnement et les choix opérés pour la prise en compte des diverses exigences en lien avec les aspects multienvironnements. *MANET-Sim* est une plate-forme développé en JAVA pouvant être paramétrée pour simuler des scénarios centralisés ou décentralisés faisant intervenir plusieurs centaines d'agents embarquant des MRC et pouvant être confronté à des agents malicieux. *MANET-Sim* est un simulateur conçu pour offrir la possibilité aux utilisateurs d'ajouter à celui-ci, en plus de nouveaux MRC, de nouvelles métriques, comportement malicieux ou protocoles. Les sources et exécutables de MANET-Sim sont disponibles à l'adresse *https ://github.com/MANET-Sim*

3.1 Analyse des besoins

Dans cette section nous identifierons les besoins découlant de la modélisation des agents embarquant un modèle de réputation et/ou de confiance et évoluant au sein des

différents environnements identifiés en 2.2.2. Nous nous pencherons d'abord sur l'analyse des besoins portant sur la modélisation d'un modèle de réputation et/ou de confiance pour ensuite analyser ceux relatifs aux environnements *MANETs*, *MAS*, *P2P* et *WSN*. Nous déterminerons ensuite l'ensemble des fonctionnalités attendues du simulateur.

3.1.1 Modèle de réputation et/ou de confiance

Au cours du chapitre 2 nous avons présenté de nombreux de modèle de réputation et/ou de confiance. Si chacun d'entre eux était différent des autres par ses méthodes et mécanismes, nous avons constaté que tous les modèles répondaient à des principes généraux (section 2.1). Dès lors il est possible d'identifier deux activités principales à partir de la figure 2.1 :

1. **Calcul** : Une première activité principale de calcul consistant à produire les différentes valeurs nécessaires au fonctionnement du modèle (confiance, réputation, fiabilité ...) à partir d'information de première et/ou de seconde main et se réalisant essentiellement lors de la phase 2.
2. **Décision** : Une seconde activité de décision opérant en phase 1, 3 et 5 où le modèle à partir d'informations ou de valeurs calculé va prendre des décisions.

Ces deux activités nous renseignent sur les différents besoins d'un modèle :

- L'activité de calcul impose l'accès aux informations de première et seconde main. Si pour les premières ces informations peuvent être stockées localement, les secondes doivent être accédées directement auprès des recommendeurs et impliquer une interface de communication.
- L'activité de décision implique que le modèle peut être sollicité par l'agent qui l'héberge et lui fournir une réponse à sa sollicitation. La figure 2.1 nous apprend que le modèle sélectionne pour l'agent un candidat (généralement celui possédant le meilleur score) parmi une sélection de candidats. Aussi l'analyse des modèles *Trip* (section 2.4.2) et *Arman* (section 2.4.4), nous apprend que le modèle peut avoir un rôle de décision sur l'acceptation et/ou le traitement d'un message.

3.1.2 Environnements d'utilisation

Présenté en section 2.2.2 l'environnement MANET nous a été défini comme un réseau sans fil à topologie évolutive. L'évolutivité de la topologie étant due à la mobilité des agents présents dans le réseau, ceux-ci devant constamment maintenir à jour une liste de leur voisin. Entre autres l'aspect sans fil autorise la promiscuité entre les différents agents qui peuvent ainsi observer les comportements de leur voisin et les évaluer. On retrouve

aussi d'autres particularités apportées par la diversité des MANETs comme la présence de senseur. Ainsi de cette description nous pouvons déduire les besoins suivants :

- L'agent est mobile et se déplace de manière aléatoire ou suivant un schéma prédéfini suivant une vitesse définie (pouvant être propre à chaque agent).
- L'agent doit maintenir à jour la liste des voisins qu'il peut contacter au fil de ses déplacements.
- L'agent doit être capable de calculer l'existence ou non d'un chemin vers un destinataire.
- L'agent peut observer son voisinage grâce à la promiscuité des réseaux sans fil et évaluer les comportements.
- Le réseau MANET doit permettre la simulation de variantes, telles que les Senseur-MANET (SMANET), Véhicule-MANET (VANET) et Internet-MANET(iMANET)

La présence de l'aspect de promiscuité permet d'identifier une activité de monitoring réalisé lors de la phase 1 pour la collection d'information sur le voisinage dont le modèle est témoin au travers de l'agent. Néanmoins cette activité d'exploitation de la promiscuité doit être utilisée lors d'un scénario de test le permettant. Par exemple, dans un scénario client/serveur similaire à celui-ci de *TRM-Sim* ou de *ART* consistant à la réalisation d'un service par un serveur pour un client, un autre client se trouvant en promiscuité peut observer l'échange entre le client et le serveur (admettons qu'il est aussi demandeur du service et que les échanges se font sans chiffrement, par exemple des services web) et en profiter pour évaluer le serveur.

Entre autres la figure 2.14 nous informe sur les aspects des réseaux *MANETs* pour permettre les simulations des réseaux *WSN*, *P2P* et *MAS* :

- La suppression de la mobilité dans un *SMANET* doit permettre de simuler un réseau *WSN*.
- La suppression de la mobilité dans un *MANET* doit permettre de simuler des réseaux *P2P* et *MAS* sans fil
- La suppression de la mobilité et du sans-fil dans un *MANET* doit permettre de simuler des réseaux *P2P* et *MAS*

3.1.3 Métriques

L'analyse des métriques en section 2.5 nous a permis de les classer au tableau 3.11 en mettant en évidence différents axes que nous reformulons dans la liste des besoins suivante :

- Il doit être possible de consulter ou d'obtenir les métriques suivant deux portées, celle de l'agent ou du système.

– Il doit être possible de consulter deux types de métriques, celles propres aux modèles de réputation et/ou de confiance et celles liées aux réseaux.
– Il doit être possible d'étendre facilement des métriques existantes ou d'implémenter des nouvelles.

3.1.4 Protocoles

En section 2.7 nous avons discuté des protocoles client/serveur des simulateurs *TRM-Sim*, *ART* et *TOSim* et mis en évidence le caractère uniquement décentralisé de ceux-ci. *MANET-Sim* couvrant les simulations de réseaux centralisé et décentralisé, il est nécessaire de prévoir plusieurs protocoles de simulation suivant le type réseau que l'on souhaite simuler. Les protocoles décrivent comment durant une simulation les agents vont communiquer entre eux, leurs objectifs et leur fonctionnement. Ainsi un scénario décentralisé amènera les agents à obtenir auprès d'autres agents la réalisation de service leur procurant la plus haute satisfaction avec pour objectif de minimiser l'appel à des services de faible qualité pour maximiser leur satisfaction personnelle (illustration du protocole du simulateur *ART*). A l'inverse un scénario dit centralisé verra l'ensemble des agents coopérer ensemble pour atteindre un objectif commun (ex : acheminer des données collectées à un point central).

3.1.5 Comportements malicieux

La section 2.3 nous a permis d'approcher différents types de malicieux dans le cadre d'un contexte d'échange de message. Nous avons constaté deux grandes familles de malicieux avec d'une part les malicieux autonomes tel que les *Oscillants* et d'autres parts les malicieux oeuvrant en groupe comme les *Bad-Mouthing*. Ces constatations nous permettent de dériver les besoins suivants :
– Le simulateur doit permettre d'associer un comportement autre que le comportement normal à un agent.
– Le simulateur doit permettre facilement la création de nouveaux comportements autonome ou de groupe.
– Le simulateur doit permettre de simuler en même temps plusieurs types de comportement au sein d'un même réseau.

3.1.6 Besoins liés à la simulation

Aux besoins identifiés dans les sections précédentes, nous ajoutons d'autres besoins issus de la conception d'un simulateur.

- Le simulateur doit permettre de visualiser (graphiquement) le réseau
- Le simulateur doit permettre de consulter les métriques.
- Le simulateur doit permettre de créer des agents configurables (en termes de modèle embarqué, comportement, capacité...) et de supprimer des agents d'un réseau.
- Le simulateur doit permettre de créer manuellement ou automatique des connexions entres les agents du réseau. Ces connexions doivent représenter des connexions sans fil ou filaires entre les agents. Dans le cadre d'un environnement sans fil, les connexions représentent la portée d'émission d'un agent et peuvent être propres à celui-ci. Les aspects de mobilité couplée aux environnements sans fil imposent un établissement des connexions sans fil entre agent de manière dynamique et temps réel.
- Le simulateur doit permettre la suspension et la reprise d'une simulation.
- Le simulateur doit permettre d'exécuter des simulations préétablit durant un temps définit.
- Le simulateur doit offrir la possibilité de charger des configurations de réseaux déjà existants et stockés sous forme de fichier XML.

3.2 Conception

Nous présentons dans cette section les choix de conceptions réalisées pour répondre aux besoins de l'analyse de la section précédente. Nous débuterons par présenter la conception générale du simulateur *MANET-Sim* dont le rôle de faire évoluer ces agents possédant des configurations variées dans différents réseaux simulés. Nous introduirons ensuite la conception d'un agent configurable pour le simulateur (que nous appellerons Agent MANET-Sim).

3.2.1 Architecture de MANET-Sim

L'architecture de *MANET-Sim* (figure 3.2 se compose de 4 composants principaux :
- Plusieurs agents configurables (*Agent MANET-Sim*).
- Un gestionnaire de graphe 2D du réseau (*Graph2DMgr*) possédant une dimension fixe (ex : 600x600px). Dont le rôle est de maintenir une représentation du réseau simulé sous la forme d'un graphe en deux dimensions.
- Un gestionnaire d'agents (*AgentManager*) dont le rôle est de gérer l'ensemble des agents présent durant la simulation et faire office de le lien entre l'interface graphique et le gestionnaire de graphe

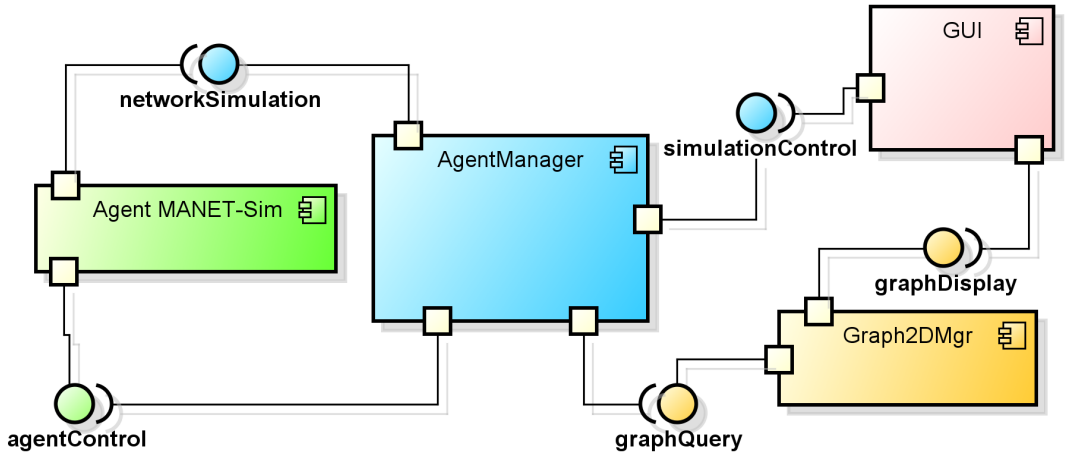

FIGURE 3.2: Architecture de MANET-Sim

– Une interface homme-machine (*GUI*) permettant de contrôler la simulation et d'afficher le graphe du réseau en temps réel

Le composant *Agent MANET-Sim* est un composant géré et crée (un par agent) par le Gestionnaire d'agent. À cet effet celui-ci fournit au Gestionnaire d'agent une interface *agentControl* permettant à ce dernier d'opérer un contrôle sur ses différents composants (ex : modifier ses paramètres, transférer des messages). Au cours d'une simulation l'agent va collaborer avec les autres agents du réseau et grâce à l'interface *networkSimulation* obtenir de la part du gestionnaire d'agent diverses informations sur sa situation telle que la liste d'agent auxquels il est connecté, des renseignements sur l'existence ou non d'un chemin vers une destination donnée, ou lui permettre d'interagir avec les autres agents du réseau. Toujours grâce à l'interface *networkSimulation* et ce dans le cadre d'une simulation de type *MANET*, chaque agent est chargé de calculer lui même ses déplacements et de fournir sa nouvelle position au gestionnaire d'agent qui répercutera celle-ci sur le graphe. Nous continuerons de détailler plus en profondeur le fonctionnement des composants Agent MANET-Sim et leur interaction avec le Gestionnaire d'agent à la section 3.2.2.

Le Gestionnaire d'agent est le composant central du simulateur MANET-Sim. À cet effet il a la possibilité de :

– Créer et supprimer des agents *MANET-Sim*
– Controler les agents (modification des configurations) par leurs interfaces *agentControl*

– Obtenir et consulter les métriques collectées par les agents par leurs interfaces *agentControl*
– Fournir de l'information aux agents sur leur situation dans le réseau et leur voisinage (via l'interface *networkSimulation*) en interrogeant le Gestionnaire de graphe 2D (interface *graphQuery*)
– Répercuter les déplacements des agents auprès du gestionnaire de graphe 2D au travers de l'interface *graphQuery*
– Calculer les connexions dynamiquement entre les agents (en fonction de leur position sur graphe et de leur portée d'émission) et les répercuter sur le graphe.

L'interface graphique vient se connecter au graphe pour réaliser la visualisation en temps réel (interface *graphDisplay*) de celui-ci, ainsi qu'au Gestionnaire d'agent pour pouvoir contacter les agents et obtenir d'eux des métriques locale (propre à l'agent) ou globale (par agrégation des données locales). L'interface graphique a aussi la possibilité de contrôler l'ensemble de la simulation (ajout/suppression d'agent, pause, changement des paramètres ...) au travers de l'interface *simulationControl*.

Le Gestionnaire de graphe 2D du réseau, figure 3.2, est représenté dans *MANET-Sim* par l'API Java JUNG [1] offrant divers outils de visualisation de graphe (figure 3.3), de création (noeud, arc unidirectionnel, bidirectionnel) ainsi que des méthodes de recherche et de manipulation de graphe (calcul de chemin, liste de voisin...). *MANET-Sim* ne proposant actuellement que la création d'arcs bidirectionnelle impliquant uniquement la création d'une connexion entre deux agents que si ceux-ci peuvent mutuellement s'atteindre.

3.2.2 Agent MANET-Sim

L'analyse des besoins portant sur les modèles de réputation et/ou de confiance et sur les différents aspects des réseaux et attendue du simulateur nous ont permis de concevoir un agent configurable composé d'une série de concept illustré à la figure 3.4 qui permettent de le caractériser et de définir son comportement générale. On retrouve :

– Un **Rôle** définissant le comportement de l'agent *par rapport aux demandes de service des autres agents*. Un agent non malicieux tentera toujours de satisfaire au maximum les demandes de service qu'il recevra, au contraire d'un malicieux qui pourra, suivant son type, plus ou moins satisfaire une partie des demandes de service. Nous définirons les rôles en référence aux différents types de malicieux présentés en section 2.3 (normal, On-Off, BM...).

1. JUNG Graph API : http ://jung.sourceforge.net/

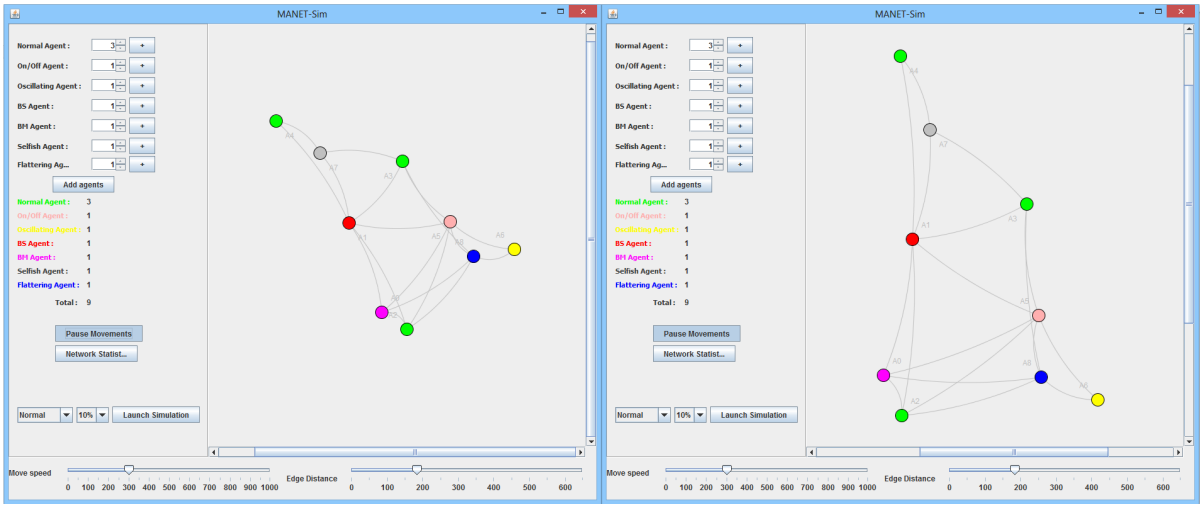

FIGURE 3.3: Visualisation d'un réseau dans MANET-Sim, fonctionnalité de zoom et de rotation grâce à l'API JUNG

- Un **Modèle** de réputation et/ou de confiance embarqué réalisant les décisions de l'agent en termes de coopération avec les autres agents (acceptation ou rejet d'une demande de service en fonction de son niveau de confiance ou de sa réputation) et de réalisation de service (choix de candidat).
- Une **Connectivité** soit filaire ou sans-fil. Cette dernière pouvant amener l'agent à la réalisation de comportements supplémentaires et liés à des aspects comme la promiscuité (traitement et enregistrement de l'information).
- Un **Protocole** définissant *comment l'agent communique avec les autres agents et comment l'agent réalise un service demandé.*
- Une série de **Mètriques** que l'agent se charge de collecter durant toute sa durée de vie.
- Une **Mobilité** caractérisée par une vitesse de déplacement et des destinations, si la vitesse de déplacement est nulle alors l'agent est immobile.
- Un **Comportement** définissant *un ensemble d'action conditionné et réalisé par l'agent.* Un exemple de comportement produit par l'agent serait que celui-ci émette à intervalle fixe un message à un autres agent ou recherche la réalisation d'un service précis parmi son voisinage.
- Un **Objectif** définissant *la raison de vivre de l'agent* (ses buts). L'objectif est étroitement lié aux autres concepts par le fait que leur somme définit celui-ci. Un exemple d'objectif serait la maximisation de sa satisfaction personnelle (toujours

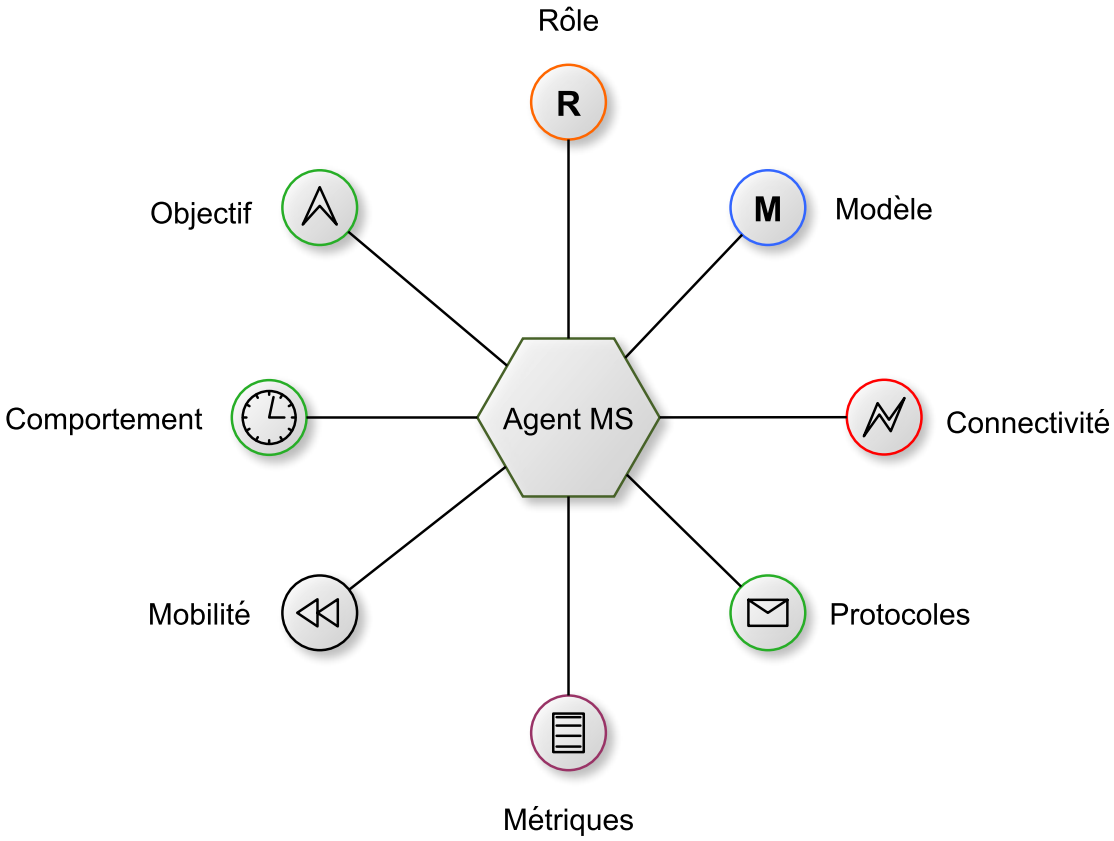

FIGURE 3.4: Concepts caractérisant un agent MANET-Sim

demander un service auprès du voisin fournissant le plus haut niveau de satisfaction) ou encore envoyer un maximum d'information collecté à un autre agent en évitant que celle-ci soit intercepté par des malicieux.

Présenté en figure 3.5, nous constituons le diagramme de composant de l'agent MANET-Sim d'une série de composants reprenant chacun un ou plusieurs des concepts illustrés précédemment.

TransactionMgr

Symbolisé en couleur verte nous retrouvons le composant *TransactionMgr* chargé d'intégrer les concepts de Rôle et de Protocoles par la gestion de la création et la réception de ce que nous appelerons des transactions au sein du système et correspondant à des

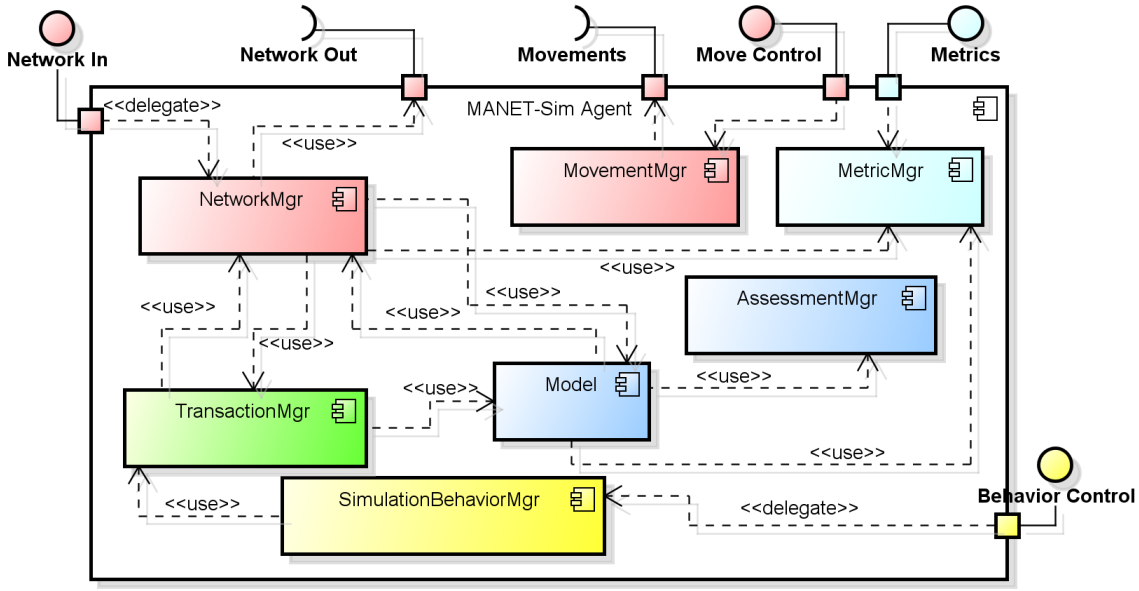

FIGURE 3.5: Diagramme de composant d'un agent MANET-Sim

paquets que peuvent s'envoyer entre eux les agents. Le gestionnaire de transaction s'occupe d'appliquer les comportements de l'agent en fonction du protocole choisit et du rôle de l'agent au traitement des transactions. Par exemple dans le cadre d'un protocole client/serveur comme dans le simulateur *ART*, un agent jouant le rôle d'un malicieux de type *On/Off* n'executera qu'une fois sur deux son protocole de réalisation de service (dans l'exemple du protocole de *ART*, cela sera la génération d'une expertise) auprès d'un demandeur. Une transaction est une structure symbolisée par le tableau 3.1 et constituée de 6 attributs :

TransactionId :string
From :string
To :string
Type :transactionType
TTL :int
Path :string[]

TABLE 3.1: Structure d'une transaction

– **TransactionId** : Un identifiant propre à l'agent créateur de la transaction, celui-ci reste inchangé jusqu'à la fin de l'utilisation de la transaction dans le réseau.

- **From** : L'identifiant sur le réseau de l'émetteur de la transaction.
- **To** : L'identifiant sur le réseau du destinataire de la transaction.
- **Type** : Élément symbolisant un type de transaction (ex : message, alerte...)
- **TTL** : *Time-To-Live*, symbolise la durée de vie en nombre de saut de la transaction (un saut est considéré comme le déplacement de la transaction vers un autre agent) . À chaque saut la durée de vie de la transaction est décrémentée d'une unité jusqu'à atteindre zéro. Une fois à zéro le relais d'une transaction est arrêté.
- **Path** : Contient la liste des identifiants des précédents agents ayant fait office de saut, ce mécanisme est utilisé pour éviter l'emprisonnement de transaction dans des routes cycliques.

L'utilisation des transactions sera présenté plus en détail dans la section 3.3.1 relatif aux protocoles de simulation.

NetworkMgr

Le composant *NetworkMgr* fait office de gestionnaire réseau implémentant le concept de Connectivité pour l'agent au travers de différentes interfaces (entrée et sortie réseau) lui permettant de communiquer avec le *Gestionnaire d'agents* et d'informer en temps réel son agent sur la liste des voisins contactables, l'existence de chemin vers un destinataire donné, d'obtenir ou de fournir de l'information auprès d'un recommandeur ou encore d'émettre et de recevoir des transactions. Le gestionnaire de réseau est aussi chargé, dans le cadre d'un réseau sans fil, de réaliser la simulation de la promiscuité auprès des autres agents (voir section 3.3.3 pour plus d'informations).

MovementMgr

Le composant *MovementMgr* est chargé de simuler la mobilité de l'agent (concept de Mobilité) dans son environnement et d'en réaliser le déplacement dans la cadre de simulation *MANETs*. Les déplacements se présentent sous la forme d'une coordonnée vers laquelle l'agent va se rendre à une vitesse prédéfinie (en nombre de pixels par seconde), une fois la destination atteinte l'agent attendra un temps définit avant d'atteindre une nouvelle destination. Le composant *MovementMgr* peut générer lui même aléatoirement les destinations ou lire celles-ci à partir d'un fichier, les mouvements réalisés par l'agent sont communiqués au travers de l'interface *Movements* au *Gestionnaire d'agents* qui se charge de créer et de supprimer dynamiquement les connexions entre les agents en fonction de la distance de connexion maximale exprimé en pixel de chaque agent. L'interface *Move Control* permet de modifier en temps réel les différents paramètres de l'agent

en terme de mobilité comme sa vitesse de déplacement, son temps d'attente, sa portée d'émission ou encore de choisir de l'immobiliser.

Model & AssessmentMgr

Le composant *Model* embarque le modèle de réputation et/ou de confiance dont le rôle est d'appuyer, en terme de décision, notamment sur ses choix de candidat (concept de Modèle). A cet effet le composant modèle peut avoir recours au *NetworkMgr* pour obtenir de l'information de seconde main auprès de recommandeurs ou utiliser l'*AssessmentMgr* (gestionnaire d'évaluation) pour obtenir les informations de première main qu'il a collectées durant la simulation. Le composant *AssessmentMgr* étant l'équivalent local d'une base de données dans laquelle l'agent va collecter et consulter les différentes évaluations qu'il réalisera.

SimulationBehaviorMgr

Le composant *SimulationBehaviorMgr* (concept de Comportement) est un composant chargé d'orchestrer le comportement de l'agent en termes d'émission de transaction (ex : fréquence, destinataire) lors d'une simulation de type *Scénario* (le mode de simulation *Scénario* est présenté plus en détail en section 3.3.2). L'interface *Behavior Control* fournie par le composant permet de paramétrer celui-ci lors de l'initialisation de l'agent ou durant une simulation. Ainsi lorsqu'une transaction est généré par le *SimulationBehaviorMgr*, celle-ci est ajouté à la file d'attente du Gestionnaire de transaction (*TransactionMgr*) qui la traitera selon le principe *FIFO* (*First in, first out*).

MetricMgr

Le composant *MetricMgr* est chargé de conserver les différentes métriques collectées et d'en permettre la consultation grâce à l'interface *Metrics*. Ce composant reçoit des métriques propres aux réseaux par le *NetworkMgr* et propres aux modèles de réputation et/ou de confiance directement par le composant *Model*.

3.3 Implémentation

Dans cette section nous présentons l'implémentation des différents protocoles et modes de simulation du simulateur. Nous détaillons aussi l'implémentation des modèles, rôles et métriques et discutons des possibilités d'extensions de ceux-ci.

3.3.1 Protocoles

Dans cette section nous présentons les différents protocoles implémentés dans *MANET-Sim* et permettant la réalisation de scénarios de simulation centralisée et décentralisée. Notons que les possibilités de configuration de l'*Agent MANET-Sim* (présenté plus en détail en section 3.3.2) permettent de base des variantes ou des combinaisons de ces protocoles.

Protocole 1 : Scénario centralisé avec routage

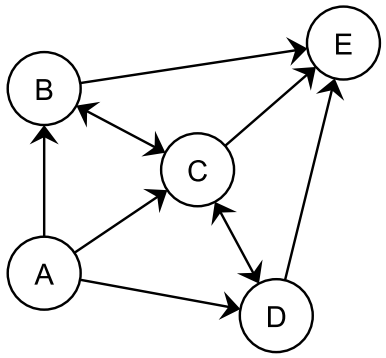

FIGURE 3.6: Illustration du Protocole 1

Le premier protocole implémenté dans *MANET-Sim* est un protocole offrant la possibilité d'un scénario centralisé se basant sur l'échange et le transfert entre les agents de paquet. La mise en place de ce protocole consiste à déterminer parmi de l'ensemble des agents présents sur le réseau un ou plusieurs agents faisant office de point de ralliement pour l'ensemble des paquets du réseau (cette opération se réalise par une paramétrisation dans ce sens du *SimulationBehaviorMgr*). Le but du réseau dans ce scénario est de maximiser le nombre de messages acheminé aux différents points de ralliement du réseau alors que le but des agents malicieux est d'empêcher leur arrivée. La mise en place d'agent en tant que point de ralliement se faisant simplement en spécifiant ceux-ci comme destinataire des transactions véhiculé par le réseau MANET-Sim. Le protocole 1 permet de simuler l'ensemble des réseaux centralisé, par exemple, ceux de type MAS illustré dans [6] et [17], mais aussi d'évaluer les capacités d'un modèle a réagir face à la découverte de malicieux sur ses routes et à assurer la continuité de ses services. Ce protocole amène les agents a réaliser l'acheminement de transactions vers des destinataires connu à l'avance. La difficulté étant de réussir à faire parvenir les transactions à ceux-ci par un chemin

fiable (pas nécessairement le plus court), à cet effet le protocole 1 se compose de deux phases :

1. **Analyse** : la première phase d'analyse intervient lorsqu'un agent reçoit une transaction. Celle-ci consiste pour l'agent a analyser la transaction qu'il a obtenue pour déterminer :
   - Si il accepte de recevoir cette transaction (évaluation de l'expéditeur par le modèle)
   - Si il est le destinataire de la transaction
   - Ou si il doit transférer la transaction à un autre agent

   Dans le deuxième cas de figure, l'agent va alors tenter de trouver un chemin vers le destinataire de la transaction et de transférer celle-ci. Cette recherche se réalisant durant la phase 2.
2. **Choix de candidat** : la seconde phase amène l'agent à étudier les différents chemins existant vers le destinataire. Dans l'illustration présentée à la figure 3.6, l'agent A doit relayer une transaction à l'intention de l'agent E, celui-ci constate la présence de plusieurs chemins lui permettant d'atteindre cet objectif : $\{B,E\}$, $\{C,E\}$, $\{D,E\}$, $\{B,C,E\}$ ... L'agent A va alors sélectionner les premiers sauts de chaque chemin existant (B,C,D) et utiliser son modèle de réputation et/ou de confiance pour choisir un candidat. Le modèle sera ainsi chargé d'évaluer les 3 agents et de ressortir un choix (ou aucun) parmi la sélection de candidats permettant de rejoindre le destinataire. Dès lors A transféra la transaction au candidat retenu par le modèle, par exemple B qui a son tour réalisera les mêmes phases que A pour délivrer la transaction à E.

Protocole 2 : Scénario décentralisé sans routage

Le protocole 2 reprend le principe d'émission de paquet du protocole 1, mais avec une limite de saut (TTL) mise à 2 et sans calcul de chemin. Dans celui-ci un agent A va transférer une transaction à un agent B dont le but sera de réussir de nouveau transférer cette transaction à un troisième agent C que B jugera digne de confiance. L'utilité du protocole est d'observer comment un modèle réussit à discerner les vrais agents réalisant un comportement malicieux (ils refusent de transférer la transaction à un tiers) des agents "incompétent" qui n'ont pas la possibilité de trouver un tiers de confiance, soit à cause d'un voisinage qu'ils détectent eux même comme étant malicieux ou à cause d'une incapacité au niveau du routage (ex : personne à qui transférer). L'aléatoire autour de la capacité de l'agent B à réussir à l'instant T permet d'ajouter un certain chaos dans le réseau en fonction de la densité de malicieux ou encore d'autres

paramètres du réseau comme les vitesses de déplacement des agents ou encore les portées d'émissions (longueur maximale des arcs dans le graphe) et d'observer la capacité d'un modèle à continuer à fournir des services ou à sombrer jusqu'à l'arrêt de tout transfert de transaction (situation de blocage du réseau). Le protocole permet aussi d'observer l'impact qu'à la promiscuité sur un modèle et si celle-ci permet une certaine "rédemption" des agents jugés incompétents (maintient d'un niveau de service acceptable malgré le chaos) ou si au contraire elle produit l'effet inverse (accentuation du chaos). Certains modèles comme Confidant n'agrègent pas de la même manière les évaluations positives et négatives et donnent plus de poids à ces dernières. Ainsi dans un environnement sans fil de forte densité (les agents sont interconnectés) la réalisation d'une faute par un agent F au sein du réseau sera perçue par un grand ensemble d'observateurs qui à leur tour deviendront des recommandeurs apportant ainsi lors du calcul d'une valeur de réputation/ confiance une forte opinion.

Protocole 3 : Scénario décentralisé client/serveur

Le protocole 3 est un simple portage du protocole client/serveur issu de *TRM-Sim*, *TOSim* et *ART*. Dans celui-ci chaque agent agit pour lui même et tente de se faire réaliser par d'autres agents le plus grand nombre de services. Les avantages de ce protocole sont d'offrir une grande stabilité du réseau contrairement au protocole 2. Cette stabilité est garantie par le fait que dans le cadre de ce protocole, seuls les agents malicieux fourniront un service de mauvaise qualité et seront potentiellement détecté par le modèle. Ce protocole permet de mettre en avant la précision du modèle dans la détection ou le rejet des agents malicieux.

3.3.2 Mode de simulation

Pour faciliter la création de réseau dans *MANET-Sim* nous avons décidé d'instaurer dans le simulateur deux modes de simulations qui sont le mode libre et le mode scénario.

Simulation libre

Le mode de simulation libre offre la possibilité à l'utilisateur de construire le réseau de son choix à partir du panel de gestion illustré en figure 3.7. Grâce à celui-ci l'utilisateur peut paramétrer le type d'agent qu'il souhaite insérer dans le réseau. On retrouve dans cette configuration la possibilité de :

– Sélectionner le modèle de réputation et/ou de confiance que l'agent va embarquer. Plusieurs modèles peuvent cohabiter dans le même réseau, mais certains d'entre

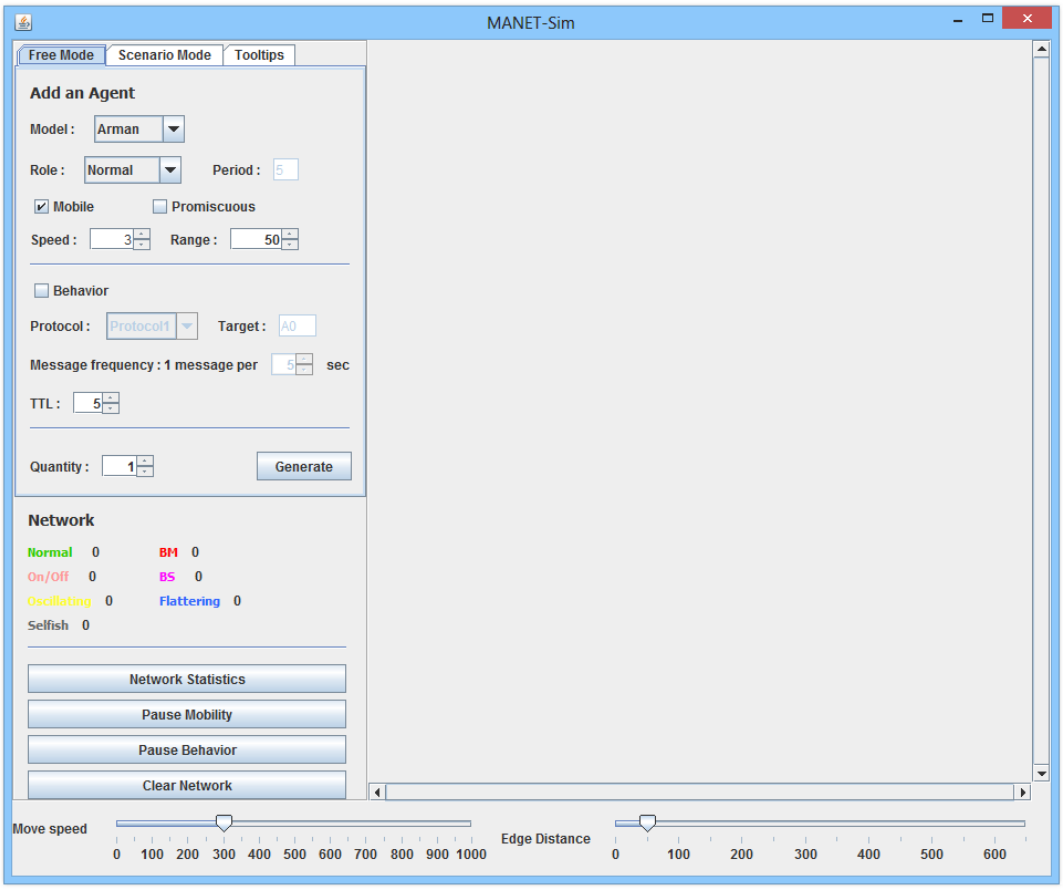

FIGURE 3.7: Interface utilisateur du mode libre

eux ne s'échangent pas le même type d'information (collection d'évaluation vs valeur précalculée) et peuvent ne pas réussir à s'échanger de l'information.

- Spécifier si l'agent joue un rôle de malicieux de type oscillant, Bad-mouthing, Ballot-stuffing ou Flattering. Et dans le cadre d'un agent de type oscillant spécifier sa période pour pouvoir simuler des malicieux de type Selfish et On/Off.
- Définir si l'agent est mobile ou non. À cet effet s'il existe un fichier de mouvement associé à l'identifiant de l'agent celui-ci réalisera les mouvements spécifiés dans celui-ci. Une fois la totalité des mouvements réalisés ou si le fichier n'existe pas l'agent se déplacera aléatoirement.
- Définir la vitesse de déplacement ainsi que la portée d'émission maximum de l'agent.
- L'activation ou non de la promiscuité.

– De définir un comportement ou non consistant à émettre à un intervalle fixe des transactions à un agent définit (protocole 1), à un destinataire parmi le voisinage (protocole 3) ou n'existant pas dans le réseau (protocole 2). L'utilisateur fixe aussi la durée de vie en nombre de saut (TTL) des transactions émissent, par l'agent. *MANET-Sim* propose aussi à l'utilisateur de sélectionner des paramètres pré définis correspondant aux protocoles présentés en section 3.3.1.

L'utilisateur peut aussi à tout moment :

– Arreter ou redémarrer la mobilité de l'ensemble du réseau ou d'un seul agent. Seuls les agents configurés comme étant mobiles reprendront leur route.
– Mettre en pause ou relancer les comportements réalisés par l'ensemble des agents ou un agent spécifique. Cette fonctionnalité n'a naturellement effet que sur les agents configurés avec un comportement.
– Modifier la vitesse de déplacement ou la portée d'émission d'un agent ou de l'ensemble du réseau.
– Supprimer un ou l'ensemble des agents du réseau.
– Créer et supprimer des connexions entre les agents.
– Déplacer manuellement un agent.
– Consulter les métriques locales d'un agent ou du réseau entier.

Mode scénario

Le mode scénario reprend les fonctionnalités du mode libre, mais propose à l'utilisateur de créer automatique un réseau à partir du chargement d'un fichier XML. Dans celui-ci l'utilisateur spécifie en plus des paramètres présentés dans le mode libre une durée maximale de simulation après laquelle *MANET-Sim* stoppera le fonctionnement entier du réseau (déplacement et comportement), laissant la main à l'utilisateur pour la consultation des métriques local ou global.

Nous présentons dans les listes suivante la structure DTD du document XML ainsi qu'un exemple d'un réseau mobile composé de 5 agents normaux, 3 agents On/Off et 2 agents Bad-Mouthing exécutant le modèle *Confidant* dans le cadre du protocole 2.

```
1 <!ELEMENT network (agent)*>
2 <!ATTLIST network
3     timer CDATA #REQUIRED
4     id CDATA #REQUIRED
5   >
6
7 <!ELEMENT agent (behavior|role|model)*>
8 <!ATTLIST agent
```

```
9       mobile CDATA #REQUIRED
10      promiscuous CDATA #REQUIRED
11      edge_length CDATA #REQUIRED
12      speed CDATA #REQUIRED
13      quantity CDATA #REQUIRED
14    >
15
16 <!ELEMENT model EMPTY>
17 <!ATTLIST model
18      name CDATA #REQUIRED
19    >
20
21 <!ELEMENT role (collusive|oscillating)*>
22 <!ATTLIST role
23      normal CDATA #REQUIRED
24    >
25
26 <!ELEMENT behavior (param)*>
27
28 <!ELEMENT param EMPTY>
29 <!ATTLIST param
30      protocol CDATA #REQUIRED
31      target CDATA #REQUIRED
32      TTL CDATA #REQUIRED
33      frequency CDATA #REQUIRED
34    >
35
36 <!ELEMENT oscillating EMPTY>
37 <!ATTLIST oscillating
38      period CDATA #REQUIRED
39    >
40
41 <!ELEMENT collusive EMPTY>
42 <!ATTLIST collusive
43      type CDATA #REQUIRED
44    >
```

Listing 3.1: Fichier DTD des configurations de réseaux MANET-Sim.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <network id="MANET01" timer="100000">
3   <agent quantity="5" speed="300" edge_length="40" promiscuous="Y" mobile="
        Y" >
4     <model name="confidant" />
5     <role normal="Y" />
```

```
6       <behavior>
7           <param frequency="500" TTL="4" target="none" protocol="1" />
8       </behavior>
9     </agent>
10
11    <agent quantity="3" speed="300" edge_length="40" promiscuous="Y" mobile="
          Y" >
12      <model name="confidant" />
13      <role normal="N">
14        <oscillating period="1" />
15      </role>
16      <behavior>
17          <param frequency="500" TTL="4" target="none" protocol="1" />
18      </behavior>
19    </agent>
20
21    <agent quantity="2" speed="300" edge_length="40" promiscuous="Y" mobile="
          Y" >
22      <model name="confidant" />
23      <role normal="N">
24        <collusive type="BM" />
25      </role>
26      <behavior>
27        <param frequency="500" TTL="4" target="none" protocol="1" />
28      </behavior>
29    </agent>
30  </network>
```

Listing 3.2: Fichier de configuration d'un réseau de type MANET constitué de 10 agents executant le protocole 2 et embarquant le modèle Confidant. Chaque agent est configuré pour émettre une transaction toutes les demi-secondes et accepter les informations provenant de la promiscuité.

3.3.3 Mécanisme d'évaluations et promiscuité

Dans *MANET-Sim* nous avons choisi de mettre en place un mécanisme pour permettre aux agents de s'évaluer entre eux et compatible avec les trois protocoles présenté en section protocole. Le mécanisme retenu à été de réutiliser les transactions échangées par le réseau en ajoutant à celles déjà existantes deux nouveaux types de transaction s'inspirant des messages d'accusé et de non-accusé de réception utilisé dans des protcoles de contrôle de transmission de paquet comme rdt [2]. On retrouve les transactions de type

2. Reliable Data Transfer

ACK et NOACK.

- **ACK** : les transactions de type ACK (Acknowledgement) sont émises par un prestataire de service (agent réalisant un service demandé) auprès d'un demandeur du service lorsque celui-ci (le prestataire) réalise parfaitement le service demandé (relais de message ...).
- **NOACK** : les transactions de type NOACK (No-Acknowledgement) sont émises par le prestataire à destination du demandeur lorsque la réalisation du service demandé n'a pu être réalisée.

Le choix de ce mécanisme simple a été motivé pour sa légèreté et son intégration optimale dans le traitement de celui-ci par les agents du système. Les nouvelles transactions venant s'ajouter à la liste d'attente du gestionnaire de transaction et permettant un traitement sans interruption des autres activités de l'agent (l'agent analyse le type de la transaction et la traite en fonction). Aussi dans le cadre de réseau sans fil l'activant, la promiscuité se réalise en deux étapes lorsque l'agent demandeur du service reçoit le résultat de sa demande de service auprès du prestataire (ACK ou NOACK). Ainsi et quelque soit le résultat le demandeur va en première étape identifier les agents en état de promiscuité par rapport à l'intéraction qu'il vient de réaliser (pour être en état de promiscuité il faut être à la fois connecté au demandeur et au prestataire) pour ensuite en seconde étape faire parvenir des copies de l'ACK ou du NOACK qu'il aura obtenu aux agents en état de promiscuité par rapport à son intéraction. Ces deux étapes sont réalisés par le composant *NetworkMgr* de l'agent demandeur du service.

Ensuite la réception d'une transaction de type ACK ou NOACK par un agent déclenchera une procédure d'évaluation auprès du modèle dont le résultat sera l'enregistrement par l'agent d'une évaluation dont la nature (positive, négative) dépendra soit du type de transaction (ACK ou NOACK) ou de l'évaluation d'une valeur (valeur de satisfaction) contenu dans la transaction ACK/NOACK.

From :string	To :string	Tmsp :timestamp	Success :boolean	Value :double

Table 3.2: Structure d'une évaluation

Notons aussi qu'une évaluation générée et stockée par l'*AssessmentMgr* est une structure présentée dans le tableau 3.2 et composée de 4 attributs :

- **From** : L'identifiant sur le réseau de l'agent auteur de l'évaluation.
- **To** : L'identifiant sur le réseau de l'agent destinataire de l'évaluation.
- **Tmsp** : Date et heure de la création de l'évaluation. Certains modèles utilisent ces valeurs pour attribuer de l'ancienneté aux évaluations.

– **Success** : Valeur booléen prenant la valeur *Vrai* si l'évaluation est positive, *Faux* si elle est négative.
– **Value** : Attribut contenant une valeur numérique. Cet attribut est utilisé par certain modèle comme alternative aux valeurs booléen.

3.3.4 Modèle de réputation et/ou de confiance

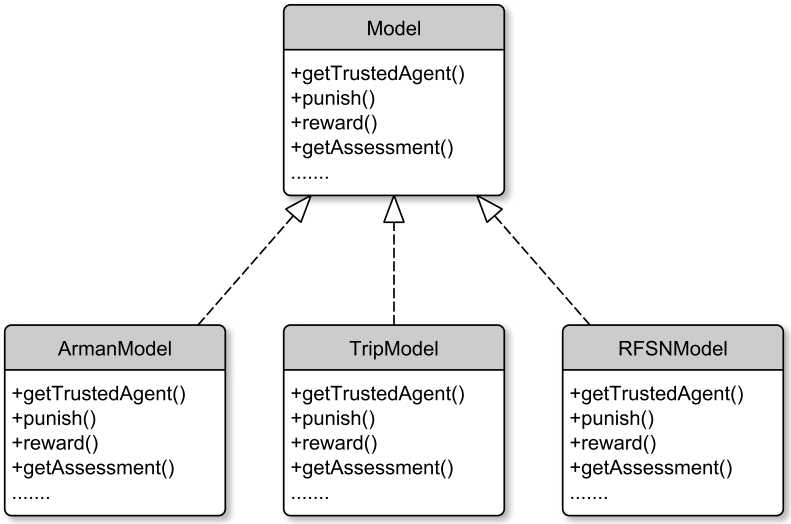

FIGURE 3.8: Diagramme de classe des classes relatif aux modèles de réputation et/ou de confiance

L'implémentation d'un nouveau modèle de réputation et/ou de confiance se réalise en deux phases. La première phase consiste à la création d'un nouveau modèle sous la forme d'une nouvelle classe JAVA implémentant l'interface *Model* de *MANET-Sim* (figure 3.8). Cette interface est constituée des méthodes :

– STRING SELECTTRUTABLECANDIDAT(ARRAYLIST<STRING> CANDIDATLIST) : Cette méthode passe en paramètre au modèle une liste d'identifiant d'agent parmi lesquels il doit réaliser une sélection d'un ou aucun candidat (généralement le candidat possédant le plus haut niveau de confiance ou la plus haute réputation). Cette méthode réalise les phases de Collecte d'Information, Score et Classement et Choix de candidat de la figure 2.1. Elle est appelée par le Gestionnaire des transactions lorsque celui-ci doit dans le cadre de l'exécution d'un protocole choisir un candidat parmi une liste de candidat fournissant des services dont l'agent est demandeur.

- void punish(String agentID) : Cette méthode passe en paramètre l'identifiant d'un agent n'ayant pas satisfait à la réalisation d'un service et demande au modèle de le punir. Cette méthode réalise la phase Punission de la figure 2.1. Elle est initiée par le Gestionnaire des transactions lorsque celui-ci reçoit une transaction de type NOACK.
- void reward(String agentID) : Cette méthode fournit l'identifiant d'un agent ayant correctement satisfait la réalisation d'un service et demande au modèle de le récompenser. Elle réalise la phase Récompense de la figure 2.1 et est appelée par le Gestionnaire des transactions lorsqu'il reçoit une transaction de type ACK.
- boolean isAcceptable(String agentID) : Comme pour la méthode *isRelayable*, le modèle est interrogé pour statuer sur l'acceptation des informations provenant d'un agent symbolisé par son identifiant. Cette méthode satisfait le besoin du modèle d'avoir un regard sur le traitement de l'information (transaction) émis en section 3.1. Elle est appelée par le Gestionnaire des transactions à chaque fois qu'il reçoit une nouvelle transaction autre que de type ACK ou NOACK pour obtenir l'avis du modèle sur l'acceptante ou non d'une transaction provenant de cet agent.
- boolean isRelayable(String agentID) : Cette méthode demande au modèle de statuer si l'agent (symbolisé par son identifiant) et suffisamment fiable pour réaliser le relais de sa transaction. Cette méthode est aussi appelée par le Gestionnaire des transactions comme lors de la méthode *isAcceptable* mais pour obtenir cette fois-ci l'avis du modèle sur la réalisation du relais de cette transaction. On retrouve ici la différence entre l'acceptante d'une information (méthode *isAcceptable*) et le transfert de cette information aux autres agents (méthode *isRelayable*).
- ArrayList<Assessment> getAssessmentList(String target) : Cette méthode est appelée par le Gestionnaire du réseau (*NetworkMgr*) lorsque celui-ci reçoit de la part d'un autre agent une demande d'obtention des évaluations possédée par l'agent sur une cible donnée (target). Le modèle est alors chargé de fournir une liste des évaluations et leurs résultats qu'il a réalisés sur la cible demandée. Cette méthode permet a l'agent de fournir des recommandations (information de seconde main) aux autres agents.

Une fois le nouveau modèle conçut, l'utilisateur n'a plus qu'à ajouter celui-ci dans la *Fabrique d'Agents MANET-Sim* présents dans le Gestionnaire d'agent pour pouvoir ajouter à ses réseaux des agents embarquant le nouveau modèle.

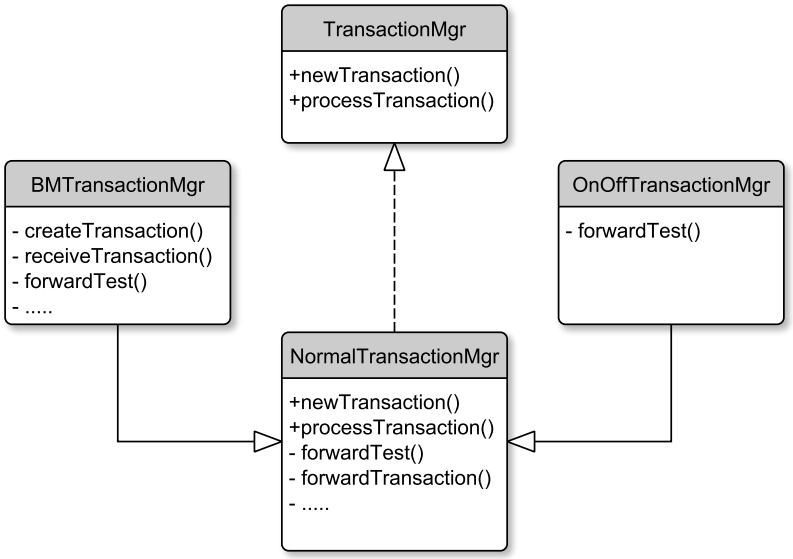

FIGURE 3.9: Diagramme de classe des classes relatif aux Gestionnaires de transactions

3.3.5 Comportements malicieux (Rôles)

L'implémentation de nouveaux comportements malicieux peut se réaliser de deux manières différentes (figure 3.9) :

- La première consiste à créer une nouvelle classe JAVA implémentant l'interface *TransactionMgr* et d'orchestrer les méthodes pour implémenter un protocole.
- La seconde consiste à étendre une classe JAVA implémentant déjà l'interface *TransactionMgr* ainsi qu'un ou plusieurs protocoles pour modifier certaines parties des protocoles pour générer un nouveau rôle.

Pour permettre l'intégration de différents protocoles, l'interface du *TransactionMgr* a été implémentée dans *MANET-Sim* en respectant le design pattern du *Template Method Pattern*. Le principe de ce pattern consiste à attribuer à une classe une méthode publique principale et de composer son exécution par une succession d'exécutions de méthodes privées qui seront traitées suivant le protocole choisit. Lorsque le gestionnaire de transaction reçoit une nouvelle transaction à traiter, celle-ci est mise en file d'attente (méthode NEWTRANSACTION(TRANSACTION T)) avant d'être traitée par la méthode principale VOID PROCESSTRANSACTION(TRANSACTION T). Les auteurs de nouveaux Gestionnaires de transactions sont alors libres d'implémenter le contenu de la méthode traitant les transactions selon les besoins du protocole qu'ils souhaitent intégrer. Dans *MANET-Sim* nous avons réalisé l'implémentation d'un Gestionnaire de Transaction pou-

vant traiter les trois protocoles que nous avons définis en section 3.3.1 et réalisant ceux-ci selon un rôle d'agent non malicieux (normal). Une fois ce Gestionnaire de transaction définit nous avons étendu (par extension de la classe) celui-ci aux différents rôles d'agents malicieux que nous avons présentés en section 2.3.

Dès lors une fois un nouveau Gestionnaire de transactions définit, celui-ci doit être, comme un nouveau modèle, ajouté à la *Fabrique d'Agents MANET-Sim* du Gestionnaire d'agent.

3.3.6 Métriques

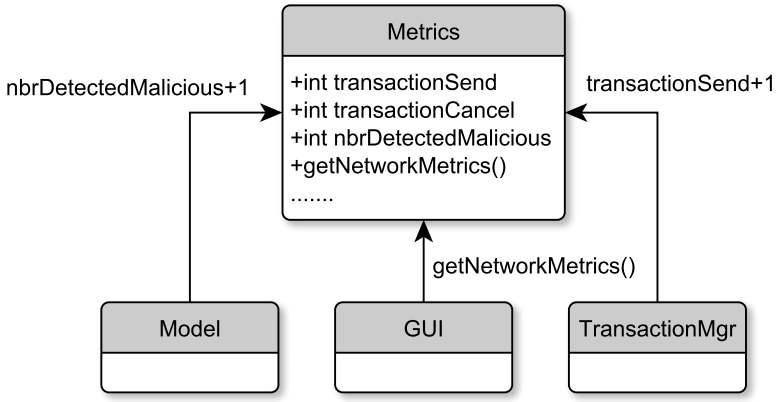

FIGURE 3.10: Diagramme de classe des classes relatif aux mètriques

L'ajout et la consultation de nouvelles métriques (figure 3.10) se réalise par l'ajout de celle-ci directement dans la classe *Metrics* embarqué par un agent (sous forme de variable manipulable au travers de celle-ci). Cette classe étant accéssible depuis le modèle ou le gestionnaire de transaction il devient alors possible de modifier les classes de ces deux composants pour rendre possible la collecte de ces données. Celles-ci pouvant être ensuite consulté depuis l'interface graphique du simulateur. Nous présentons dans les sections suivantes plusieurs métriques implémenté dans *MANET-Sim*.

Valeur de réputation, confiance, fiabilité, poids et source des évaluations

Ces différentes métriques liées au modèle de réputation et/ou de confiance sont propres à chaque agent lors du calcul d'une valeur de réputation ou de confiance. Nous avons souhaité dans MANET-Sim offrir la possibilité d'enregistrer un historique de ces

valeurs et de fournir depuis l'interface une visualisation de celle-ci. Le simulateur fournit au travers du modèle des méthodes de collectes de ces valeurs lié au calcul de la réputation/confiance d'un agent. Ainsi à n'importe quel instant de la simulation un observateur peut consulter l'évolution des valeurs calculée pour un agent (Figure 3.11) et connaitre la provenance des évaluations (directe ou indirecte) qui ont servi aux calculs de ces valeurs.

Malgré le fait que de nombreux auteurs utilisent ces valeurs pour comparer des modèles, nous trouvons que celles-ci ont un réel intérêt pour l'amélioration et l'observation des modèles, car elles offrent la possibilité de suivre pas à pas le raisonnement d'un agent dans un environnement contrôlable.

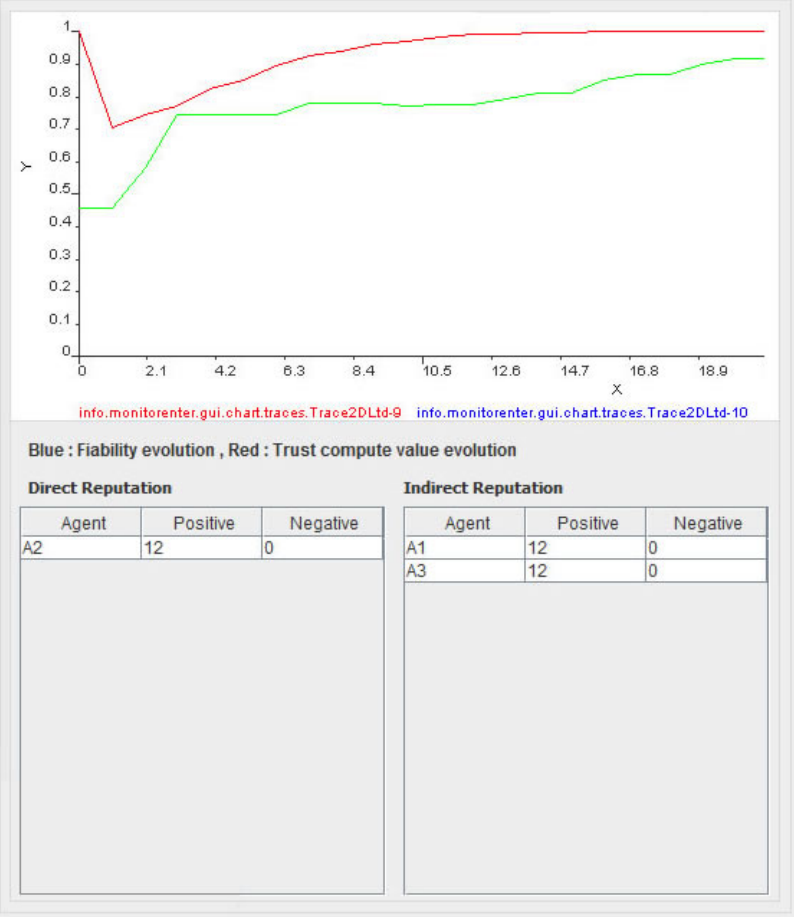

FIGURE 3.11: Évolution de la réputation (courbe rouge) et de la fiabilité (courbe verte) calculées par l'agent A2 par rapport à A0 par le modèle Arman au cours d'une simulation et détail des provenances des évaluations

Nombre total d'évaluations échangées

Cette métrique du système consiste à recenser le nombre total d'évaluations que des agents ont échangés entre eux au cours d'une simulation. Elle permet de traduire la logique des modèles de réputation et/ou de confiance dont le rôle au sein du système est de diriger l'agent vers des intervenants de confiance sans prendre en compte l'aspect d'optimisation de route (le modèle tentera d'atteindre l'objectif de l'agent en empruntant le "chemin/procédé" le plus respectable ou digne de confiance). Nous proposons cette métrique comme une sorte d'indice de consommation du système en ressource pour atteindre un certain niveau de détection ou de précision.

Détection et matrice de détection

Nous avons vu en section 2.4 que de nombreux modèles avaient recourt à des testes pour détecter les agents malicieux et les écarter. Dans MANET-Sim nous avons décidé de recenser cette détection au sein du modèle de chaque agent (lorsque le modèle embarqué le permet) et comptabiliser les bonnes et les mauvaises détections sous la forme d'une matrice de détection (Tableau 3.3). La matrice de détection d'un système est actualisée à chaque fois qu'un agent du système réalise une détection. Cette détection est enregistrée dans la matrice en prenant en compte le rôle de l'agent auteur de la détection et le rôle de l'agent détecté lors de la détection. La correspondance des deux rôles est réalisée au sein de la matrice en incrémentant d'une unité la cellule de la matrice correspondante. L'utilisation de la matrice de détection permet d'obtenir de nombreuse information au sujet de la précision du modèle. Nous pouvons par exemple obtenir la répartition des détections au sein du système ou encore établir un score total de bonne détection (détection réelle d'un agent malicieux) pour le modèle. Elle permet aussi de connaitre le pourcentage de détection d'agent incompétent qui sont des agents réalisant un comportement normal, mais détecté comme malicieux, car dans l'impossibilité de fournir des services à cause de sa situation géographique (pas de candidat au transfert) ou à cause de son environnement (voisinage détecté comme malicieux). Notons aussi que la matrice de détection peut être consultée localement pour chaque agent ou globalement pour tout le système.

Transactions capturées et annulées

Du point de vue des métriques portant sur le réseau, *MANET-Sim* propose, en plus d'une métrique comptabilisant le nombre total de transactions émissent par le système ou un agent particulier, la comptabilisation au travers du *TransactionMgr* du nombre de transactions annulé ou capturé par un agent. Par annulé nous définissons les transactions

	Normal	**On/Off**	**Oscillant**	**Selfish**	**BM**	**BS**	**Flattering**
Normal	33	13	8	8	14	14	12
On/Off	14	42	17	20	21	28	20
Oscillant	18	20	51	26	32	32	26
Selfish	19	11	16	42	23	20	11
BM	26	25	39	40	0	35	51
BS	1	2	1	2	1	0	3
Flattering	18	30	20	17	42	32	56

TABLE 3.3: Matrice de détection globale du modèle Trip dans un environnement à 90% malicieux

dont le transfert a été interrompue soit à cause d'une impossibilité de relais due à la situation géographique de l'agent (pas de candidat au transfert) soit à cause de la détection par l'agent de son voisinage comme étant malicieux. À l'opposé nous définissons la capture d'une transaction par l'interception de celle-ci par un agent malicieux (ex : un agent égoïste). Si la première nous renseigne sur l'efficience du système pour le transfert de transaction, la seconde nous informe sur l'efficacité ou non des mécanismes de détection et de rejet des agents malicieux par le système. Ainsi pour un modèle possédant un taux de détection, la métrique de capture nous renseigne sur la quantité de comportements malicieux à observer pour atteindre ce taux de détection et permet la comparaison en terme de rapidité d'un modèle à atteindre un niveau de détection par rapport à un autre. Entre autres la soustraction de la somme de ces deux métriques avec le nombre total de transactions permet d'obtenir le nombre total de transactions ayant correctement été relayées. Ces métriques sont toutes consultables de manière locale auprès d'un agent particulier ou de manière globale pour tout le système (somme des valeurs).

Chapitre 4

Conclusion

Au cours de ce travail, nous avons identifié les différents besoins nécessaires à la conception d'un simulateur pour les réseaux *MANETs* et mis en évidence les manquements des simulateurs existant pour la simulation de ces réseaux. Pour combler ce manquement, nous avons proposé notre propre solution de simulation pour les réseaux *MANETs*. *MANET-Sim* se positionne comme un simulateur de modèle de réputation et/ou de confiance multi environnement et multi protocoles. Embarquant des fonctionnalités de visualisation et de construction des réseaux, *MANET-Sim* invite ses utilisateurs à comparer, tester, adapter des modèles dans des environnements d'utilisation variés allant de réseaux a topologies filaires jusqu'au cas extrême de mobilité sans fil. Le simulateur propose une architecture modulaire permettant la réalisation facile de configuration d'agent grâce à la mise en place d'une fabrique d'agents offrant la possibilité de concevoir sur-mesure (modèle, comportement, caractéristique) les agents d'un réseau. *MANET-Sim* propose aussi aux utilisateurs d'étendre facilement sa bibliothèque de comportement malicieux, de protocoles, de métrique ainsi que de modèles. En plus de l'aspect mobilité, *MANET-Sim* offre la possibilité de confronter les modèles à un aspect des réseaux sans fil trop souvent négligé qu'est la promiscuité.

Nous maintenons que la promiscuité est un aspect des réseaux sans fil qui ne doit pas être ignoré et que la prise en compte de celle-ci par les modèles ne peut être que bénéfique en termes de préservation de ressource pour l'agent. Cette prise en compte pourrait être encore plus bénéfique aux agents évoluant dans des réseaux mobiles où ceux-ci pourraient se contenter d'observer et se faire une opinion de leur voisinage avant d'entreprendre des collaborations et ainsi préserver au maximum leurs ressources. Néanmoins son utilisation de son utilisation pose la question de *quand* pouvons-nous agréger ces données provenant d'observations par promiscuité de comportements d'autres agents du système ?

La réalisation du simulateur *MANET-Sim* présenté dans ce travail est le fruit de plusieurs évolutions de celui-ci. D'abord conçu comme un outil de test basique pour des réseaux d'agent mobiles, *MANET-Sim* s'est vu intégrer par la suite les aspects de promiscuité, de visualisation et d'extension de ses bibliothèques de comportements et de modèles. Si celui-ci se positionne comme un outil parfaitement exploitable pour de petits réseaux (plusieurs centaines d'agents), ses performances sont fortement impactées par la visualisation graphique malgré une très grande optimisation de celle-ci. Nous pensons dans les prochaines versions fournir la possibilité de désactiver et de réactiver la visualisation graphique lors des simulations en mode scénario.

Bibliographie

[1] Amazon auctions, http ://auctions.amazon.com.

[2] ebay, http ://www.ebay.com.

[3] Karl Aberer and Zoran Despotovic. Managing trust in a peer-2-peer information system. In *Proceedings of the tenth international conference on Information and knowledge management*, CIKM '01, pages 310–317, New York, NY, USA, 2001. ACM.

[4] Eytan Adar and Bernardo A.. Huberman. Free riding on gnutella. In *First Monday [Online]*, Volume 5, October 2000.

[5] Lokesh Bajaj, Mineo Takai, Rajat Ahuja, Ken Tang, Rajive Bagrodia, and Mario Gerla. Glomosim : A scalable network simulation environment. *UCLA Computer Science Department Technical Report*, 990027 :213, 1999.

[6] Jonathan Blangenois, Guy Guemkam, Christophe Feltus, and Djamel Khadraoui. Organizational security architecture for critical infrastructure. *To appears in FARES Workshop 2013*, 2013.

[7] A. Boukerch, L. Xu, and K. EL-Khatib. Trust-based security for wireless ad hoc and sensor networks. *Computer Communications*, 30(11 -12) :2413 – 2427, 2007. Special issue on security on wireless ad hoc and sensor networks.

[8] Sonja Buchegger and Jean-Yves Le Boudec. A robust reputation system for p2p and m bile ad hc netw rks p2p and mobile ad-hoc networks. 2004.

[9] Javier Carbo, Jose M Molina, and Jorge Davila. Comparing predictions of sporas vs. a fuzzy reputation system. In *3rd International Conference on Fuzzy Sets and Fuzzy Systems*, volume 200, pages 147–153, 2002.

[10] C. Castelfranchi and R. Falcone. Trust is much more than subjective probability : mental components and sources of trust. page 10 pp. vol.1, jan. 2000.

[11] Jin-Hee Cho, A. Swami, and Ing-Ray Chen. A survey on trust management for mobile ad hoc networks. *Communications Surveys Tutorials, IEEE*, 13(4) :562–583, Quarter.

[12] Sanjay Kumar Dhurandher, Sudip Misra, Mohammad S Obaidat, and Nidhi Gupta. An ant colony optimization approach for reputation and quality-of-service-based security in wireless sensor networks. *Security and Communication Networks*, 2(2) :215–224, 2009.

[13] Karen K Fullam, Tomas B Klos, Guillaume Muller, Jordi Sabater, Andreas Schlosser, Zvi Topol, K Suzanne Barber, Jeffrey S Rosenschein, Laurent Vercouter, and Marco Voss. A specification of the agent reputation and trust (art) testbed : experimentation and competition for trust in agent societies. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 512–518. ACM, 2005.

[14] Diego Gambetta. *Trust : Making and Breaking Cooperative Relations.* Blackwell Publishers, February 1990.

[15] Saurabh Ganeriwal, Laura K Balzano, and Mani B Srivastava. Reputation-based framework for high integrity sensor networks. *ACM Transactions on Sensor Networks (TOSN)*, 4(3) :15, 2008.

[16] Félix Gómez Mármol and Gregorio Martínez Pérez. Trip, a trust and reputation infrastructure-based proposal for vehicular ad hoc networks. *Journal of Network and Computer Applications*, 35(3) :934–941, 2012.

[17] Guy Guemkam, Jonathan Blangenois, Christophe Feltus, and Djamel Khadraoui. Metamodel for reputation based agents system - case study for electrical distribution scada design. *To appears in SIN 2013*, 2013.

[18] Guy Guemkam, Djamel Khadraoui, Benjamin Gâteau, and Zahia Guessoum. Arman : Agent-based reputation for mobile ad hoc networks. *To appears in PAAMS 2013*, 2013.

[19] Bryan Horling, Victor Lesser, and Regis Vincent. Multi-agent system simulation framework. In *16th IMACS World Congress*, 2000.

[20] Audun Josang, Roslan Ismail, and Colin Boyd. A survey of trust and reputation systems for online service provision. *Decision Support Systems*, 43(2) :618 – 644, 2007. Emerging Issues in Collaborative Commerce.

[21] A Köpke, M Swigulski, K Wessel, D Willkomm, PT Haneveld, TEV Parker, OW Visser, HS Lichte, and S Valentin. Simulating wireless and mobile networks in omnet++

the mixim vision. In *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*, page 71. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008.

[22] TH Lacey, RF Mills, BE Mullins, RA Raines, ME Oxley, and SK Rogers. Ripsec– using reputation-based multilayer security to protect manets. *Computers & Security*, 2011.

[23] Felix Gomez Marmol and Gregorio Martínez Pérez. Security threats scenarios in trust and reputation models for distributed systems. *computers & security*, 28(7) :545–556, 2009.

[24] Félix Gómez Mármol and Gregorio Martınez Pérez. Trmsim-wsn, trust and reputation models simulator for wireless sensor networks. In *Communications, 2009. ICC'09. IEEE International Conference on*, pages 1–5. IEEE, 2009.

[25] Félix Gómez Mármol and Gregorio Martínez Pérez. Trust and reputation models comparison. *Internet Research*, 21(2) :138–153, 2011.

[26] Félix Gomez Marmol and Gregorio Martinez Pèrez. Towards pre-standardization of trust and reputation models for distributed and heterogeneous systems. *Computer Standards and Interfaces*, 32(4) :185 – 196, 2010.

[27] S.P. Marsh, University of Stirling. Dept. of Computing Science, and Mathematics. *Formalising trust as a computational concept*. University of Stirling, 1994.

[28] Stephen Naicken, Anirban Basu, Barnaby Livingston, and Sethalat Rodhetbhai. A survey of peer-to-peer network simulators. In *Proceedings of The Seventh Annual Postgraduate Symposium, Liverpool, UK*, volume 2, 2006.

[29] Zeinab Noorian and Mihaela Ulieru. The state of the art in trust and reputation systems : a framework for comparison. *Journal of theoretical and applied electronic commerce research*, 5(2) :97–117, 2010.

[30] Jordi Sabater and Carles Sierra. Review on computational trust and reputation models. *Artificial Intelligence Review*, 24 :33–60, 2005.

[31] W T Luke Teacy, Jigar Patel, Nicholas R Jennings, and Michael Luck. Travos : Trust and reputation in the context of inaccurate information sources. *Autonomous Agents and Multi-Agent Systems*, 12(2) :183–198, 2006.

[32] Régis Vincent, Bryan Horling, and Victor Lesser. An agent infrastructure to build and evaluate multi-agent systems : The java agent framework and multi-agent system simulator. *Infrastructure for Agents, Multi-Agent Systems, and Scalable Multi-Agent Systems*, pages 102–127, 2001.

[33] Wei Wang, Guosun Zeng, and Lulai Yuan. Ant-based reputation evidence distribution in p2p networks. In *Grid and Cooperative Computing, 2006. GCC 2006. Fifth International Conference*, pages 129–132. IEEE, 2006.

[34] Xiaodong Xian, Weiren Shi, and He Huang. Comparison of omnet++ and other simulator for wsn simulation. In *Industrial Electronics and Applications, 2008. ICIEA 2008. 3rd IEEE Conference on*, pages 1439–1443. IEEE, 2008.

[35] Giorgos Zacharia, Alexandros Moukas, and Pattie Maes. Collaborative reputation mechanisms for electronic marketplaces. *Decision Support Systems*, 29(4) :371 – 388, 2000.

[36] Yan Zhang, Wei Wang, and Shunying Lü. Simulating trust overlay in p2p networks. *Computational Science–ICCS 2007*, pages 632–639, 2007.